Analogue Neuromorphic Systems

A thesis presented by

Vyacheslav Chesnokov

M.Sc.(Moscow Institute of Physics and Technology, 1991)

to

The School of Applied Sciences

in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

in the subject of Computational Neuroscience

De Montfort University

Leicester, UK

June, 2001

Abstract

This thesis addresses a new area of science and technology, that of neuromorphic systems, namely the problems and prospects of analogue neuromorphic systems. The subject is subdivided into three chapters.

Chapter 1 is an introduction. It formulates the oncoming problem of the creation of highly computationally costly systems of nonlinear information processing (such as artificial neural networks and artificial intelligence systems). It shows that an analogue technology could make a vital contribution to the creation such systems. The basic principles of creation of analogue neuromorphic systems are formulated. The importance will be emphasised of the principle of orthogonality for future highly efficient complex information processing systems.

Chapter 2 reviews the basics of neural and neuromorphic systems and informs on the present situation in this field of research, including both experimental and theoretical knowledge gained up-to-date. The chapter provides the necessary background for correct interpretation of the results reported in Chapter 3 and for a realistic decision on the direction for future work.

Chapter 3 describes my own experimental and computational results within the framework of the subject, obtained at De Montfort University. These include: the building of (i) Analogue Polynomial Approximator/Interpolator/Extrapolator, (ii) Synthesiser of orthogonal functions, (iii) analogue real-time video filter (performing the homomorphic filtration), (iv) Adaptive polynomial compensator of geometrical distortions of CRT- monitors, (v) analogue parallel-learning neural network (backpropagation algorithm).

Thus, this thesis makes a dual contribution to the chosen field: it summarises the present knowledge on the possibility of utilising analogue technology in up-to-date and future computational systems, and it reports new results within the framework of the subject. The main conclusion is that due to its promising power characteristics, small sizes and high tolerance to degradation, the analogue neuromorphic systems will play a more and more important role in future computational systems (in particular in systems of artificial intelligence).

Acknowledgments

The author acknowledges his supervisor Professor Sue Bayliss (Solid State Research Centre) for constant help in the form of scientific advice and organisational actions, his second supervisor Dr. Buckberry all the technical staff of the University (including R.G.Wright, D.Bazeley, C.Warrington and A.Woodford), all the other students in the laboratory for routine help, and De Montfort University as a whole (including Applied Sciences School staff) for giving him the opportunity to do this research.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

The project is dedicated to investigating different aspects of the problem of creation of analogue computational systems: the problem of influence of elements' precision, different schemes of self-correction, the use of the principle of orthogonality and of polynomial representation of functions.

The aim of the project was to contribute to the theory of analogue neuromorphic systems, as well as to expand the basis of purely analogue elements in order to employ the advantages of analogue technology (compactness, low power consumption, etc.) by analogue VLSI (very large scale integration) systems. Another aim of the project was to find some new applications for analogue nonlinear adaptive (neuromorphic) systems.

The objectives of the project are first to analyse the influence of imperfections of analogue neuromorphic systems on their characteristics and to suggest some approaches to solution of this problem (different self-correction systems); second to suggest schemes of different analogue adaptive nonlinear systems (and develop breadboard-models), which could find application in purely analogue neuromorphic information processing systems (such as Analogue Polynomial Approximator, Synthesiser of Orthogonal Functions, Compensator of AO nonlinear distortions, etc.).

This chapter formulates the oncoming problem of the creation of highly computationally intensive nonlinear information processing systems, such as artificial neural networks and artificial intelligence systems. It shows that analogue technology could make a vital contribution to the creation of such systems. The basic principles of creation of analogue neuromorphic systems are formulated. The importance of the principle of orthogonality for the future highly efficient complex information processing systems is emphasised.

1.1 Neural Networks as a new promising area of science and technology

Over the last two decades, interest in Artificial Neural Networks (ANN) has been growing very fast. Computational Neuroscience, which is a new area of science dedicated to Artificial Neural Networks, has now become a very well structured, developed and expanding area of science.

There are many definitions of ANN. Let us introduce the following one: let's describe by the term *Artificial Neural Networks (ANN)* the *complex adaptive nonlinear systems of information processing.* Although some other systems (like adaptive filters) also obey this definition, I think it is the most appropriate one. Let us mention here some reasons for the fast growing interest in ANN.

- **Understanding of general principles of neural information processing**: Along with Computational Neuroscience, Neurobiology has also been a very fast developing science over last decades. The stimuli for such biological researches were the collection of information about normal and abnormal (pathological) neural activity, as well as the successes in researches into the mechanisms of information processing taking place in natural neural systems. So these two sciences represent two approaches to the problem of neural processing: the fundamental approach (Computational Neuroscience) and the phenomenological approach (Neurobiology). It should be mentioned here that Neurobiology up to now gave very little understanding of neural processes taking place in brain. A number of theories and models have been created describing some electrochemical phenomena taking place in neural cell. In particular, the mechanism of neural spike generation and propagation, synaptic exchange, different neural channels, etc. were described. Nevertheless, since the neural cell is an extremely complex object, many very important properties of natural neural networks, such as the processes responsible for the learning of the neural network as a whole, are not understood yet. This is not surprising, since it is well-known in technology (as stated by John von Neumann [4]), that the complexity of description of a complex system is progressing very fast (roughly speaking in accordance with the factorial low) with increasing number

of independent parameters (degrees of freedom) of a system. The attempts to understand the processes of natural neural network function could be compared to the attempts to understand the function of a computer processor by means of direct measurement of signals propagating between transistors. I don't believe that it is a productive approach. In cases of complex systems a fundamental approach is supposed to be much more efficient, since any elementary unit of an artificial system is well-described and understood and more complex systems (based on these units) could therefore be entirely analysed and understood. So Computational Neuroscience could be useful for interpretation of experimental neurobiological results. On the other hand the natural brain is a plentiful source of ideas and inspiration for researchers in Computational Neuroscience (since the fact that some information processing system exists, even as a biological system, is a great stimulus for creation of artificial systems doing the same).

- **Architecture**: Those who have dealt with superpowerful computational systems (supercomputers) know that it is a great problem to arrange efficient functioning of multiprocessor systems such that most of the processors are involved in the calculation process. The two main principles, which are used in supercomputers are: 1) the parallelism and 2) the staging. It is easy to see that an ANN is utilising both of these principles: the parallelism is represented by multiple neurons in a neural layer whereas the staging is represented by the multilayer architecture of ANN. It is possible to show also that neural architecture (if it is real neural architecture, and not just simulation of ANN on a uni-processor or multiprocessor system) provides the fastest possible information processing. This is since results are produced by one pass of information from the inputs of a neural network (if the ANN has been taught), through several neural layers, to the output of the ANN[1]. It apears that the complexity of optimal programming of conventional multiprocessor systems is the main reason why, until now, the universal mass-production of computers has evolved in the direction of higher speed uni-processor systems rather than in the direction of multi-processor systems. Thanks to the progress in physics and microprocessor technology, this

[1] The neural architecture: Multilayer Perceptron was used as an example of ANN architecture [5]. It should be mentioned, also, that the ANN could take a lot of learning cycles and, concequently, a lot of time for learning.

evolution has been fast enough to meet customers requirements until now. At the same time, there are physical restrictions of sizes of transistors (about 0.1 μm see [6]) and, consequently, in computational power of uni-processor systems. Therefore sooner or later the problem of creation of **universal** multiprocessor systems will arise. On the other hand the problem of optimal programming of universal multiprocessor system, it seems, cannot be solved in principle. That is, it's very difficult (virtually impossible) to create for a multiprocessor system a compiler which will generate an optimal code from a higher level language (like e.g. C or Fortran). In the same way, it is impossible to create the optimisation procedure which will guarantee the convergence of the parameter of quality of an arbitrary complex nonlinear system to its **global** optimum. At the same time, the problem of optimal programming of universal multiprocessor systems must be solved (by some sub-optimal way), since it would be inappropriate if the multiprocessor system, containing let's say 10000 processors, will use on average, let's say, 100 processors.

So one of the main advantages of neural architecture is its ability to provide the maximal possible speed of information processing and supposedly near-optimal computational process.

- **Task statement**: However, to realise the potential of such a computationally powerful system, a learning procedure is required which adjusts the strength of interconnections between neurons of ANN. The ANN learning process very often takes a lot of time. At the same time the advantage of ANN is that the task for ANN could be stated on a very high "level of abstraction" (that is, could be much less formalised). On the other hand to create programs in C or other similar languages (or even in, so called, languages of artificial intelligence, like Lisp or Prolog, see [7] and [8]) a lot of work should be done by the programmer (and, probably, by a modeller) to convert real-life problems to these suitable for computer formal logical forms. So the learning system of ANN is performing the same function as a programmer+compiler of conventional computers, that is, the adaptation of real-life problems into a form suitable for a particular hardware. This is one of the reasons why ANNs are so attractive for researchers and developers of computational systems, that is their ability to perform

functions which in the case of conventional computational systems require the human's (programmer's) involvement. This property of ANNs makes them especially useful instruments for solution of problems, which are very difficult or impossible to formalise, and which very often arise in Signal Processing, sound/image processing/recognition, etc. Neural networks are very successful at the handling such kind of problems.

Both these two properties of ANN (fastest, maximally parallel architecture and "high level" of task statement) are very critical for Artificial Intelligence (AI) system creation. Therefore one of the main motivations of neural researches is the creation of the hardware basis for systems of Artificial Intelligence.

There is a very interesting question to answer: what is the complexity of the system of Artificial Intelligence? This question seems to be extremely complex and is beyond the scope of this thesis, nevertheless we can state it and explain its importance. Obviously the AI system will be simpler than the human brain (we are assuming here that AI systems will not blindly mimic the natural brain, but will be based on principles of rationality). The question is: how much simpler? There are many examples of artificial systems which are much more efficient than natural systems. We can mention here the wheel, which allows us to move very fast by car or train (whereas animals can move only step by step, which seems to be much less efficient). As for the information processing system, the simplest calculator is a good example of a very efficient (in comparison with the natural brain) high precision computational system. It takes much less time to perform the computation, consumes much less power and is much smaller in size than the human brain. This fact (irrationality of natural systems) could be explained by the fact that the human brain is the result of evolution from the simplest one-cell organism into a human. Besides, any organism (in particular, humans) develops from one cell into the whole organism. Therefore some researchers could assume that AI systems could be several orders of magnitude simpler than the natural brain. My personal opinion, however, is that the complexity of AI systems will be comparable with the complexity of the human brain. This opinion is based on analysis of problems the human brain (and hence the AI system) copes with. Our brain is irrational in cases of logical problems and high precision computation. At the same time, it seems the human brain has a proper architecture to solve the nonformalised problems as well

as to process badly conditioned and/or noisy data and so on. Actually nowadays an estimation of the computational power of the human brain has been agreed amongst researchers to be approximately 10^{11} neurons. Each neuron has on average 10^3 synapses. It could be assumed also that it takes 0.1 sec for information to propagate through the brain. So the computational power of the human brain could be estimated as 10^{15} multiplications per second. This is approximately 10^6 times higher than the computational power of a conventional uni-processor computer. So one can assume the possibility of creation of artificial computational systems with comparable computational power in the foreseeable future (10-30 years). However, the problems of huge size memory, as well as of the appropriate computational architecture are to be solved. Besides, it is very important to make such AI systems reliable, of small size and low power consumption.

One way or another, Computational Neuroscience will be an important and fast-developing branch of science and technology at least until the Artificial Intelligence systems (that is- artificial systems, which are as intelligent as a human brain) are created. This will not happen soon (maybe in a hundred or in several hundred years).

During the next few decades, the creation of systems solving different tasks of recognition, control, optimisation (in particular, the control of robots and automata) in real-time will be the most important problem for Computational Neuroscience.

In this connection it should be emphasised that the technology should not blindly follow the biology, trying to copy particular schemes and approaches. That could be extremely irrational. Let us consider the Silicon Retina (described in [6]), which, in my view, is an example of such irrational systems. The description of the Silicon Retina is in subsection 2.3.1. The Silicon Retina is an integrated structure,which is not only producing the video-signal (as the conventional CCD-based video-cameras do), but also is performing nonlinear spaced brightness equalisation. So, the overall contrast of the picture is increasing, which results in increasing the efficient dynamic range of the video signal. The suggested scheme of the Silicon Retina is mimicking the biological retina (see subsection 2.3.1). In spite of the great importance of the Silicon Retina for the interdisciplinary branch of science and technology: Neuromorphic Systems, it is easy to show its irrationality. Indeed, the Silicon Retina has a parallel computational architecture. The system is implemented on the basis of standard silicon VLSI technology, thus the speed of the nonlinear transform is greater than 10 MHz, whereas the images are normally changing very slowly (~20 Hz). Therefore the potential (i.e. the speed) of the

standard VLSI technology will not be used in normal situations. Such systems however could be used for high speed applications (for example super high-speed video recording, a very exotic problem which probably will have some importance in the future). For standard applications in real-time nonlinear image processing (compression of the dynamic range, contrast enhancement, etc.) serial (rather than parallel) systems seem to be more appropriate.

The dynamic range of the analogue signal taken from the CCD typically has very high dynamic range (more than 2000:1), 10 times higher than that of the normal digital video signal. Therefore a special microchip enhancing the analogue video signal taken from a CCD (e.g. compressing by 10 times the dynamic range of the video-signal), rather than the Silicon Retina, would be a commercial implementation of the principles of image processing used by the biological retina. In subsection 3.5 an analogue system is described which performs real-time nonlinear transform of a video-signal (the Homomorphic Filtration), similar to the nonlinear transform performed by the Silicon Retina.

Conclusion: Artificial systems of information processing could be much more computationally efficient than natural ones. The aim of my work is to contribute to the creation of complex and supercomplex computational systems, in particular ANN, which, supposedly, will be the main components of Artificial Intelligence systems. Another aim is to suggest simple and highly efficient computational systems, which could be implemented in the near future.

1.2 Analogue technology - fundamental advantages and drawbacks

1.2.1 Advantages

The human brain is the most complex and reliable information processing system known up to now. At the same time, I don't think that there is something incomprehensible in the human brain that is vital for its information processing function and yet can not be reproduced (and utilised) in artificial systems.

Although there is at present no complete understanding of informational processes taking place in the brain, there is some understanding of electro-chemical and biological processes taking place in neural cells. This allow us to analyse some of the properties of natural neural networks. These properties are for the most part the parameters of nonlinear information processing, which in the case of ANN are related to feedforward networks (the processes of *learning* by natural neurons are far less understood).

The clue to success of a Natural Neural Network (NNN) seems to be that it uses the nonlinear bio-electro-chemical phenomena taking place in neural cells to perform the mathematical transforms. In [9] Carver Mead showed that the use of nonlinear physical phenomena taking place in transistors to perform mathematical transforms could give a huge gain in power consumption (more than 10^4), a reduction in size (about 100) and an increase in speed (about 100). Therefore artificial systems of information processing, based on semiconductor VLSI technology and utilising physics to perform nonlinear mathematical transforms, will be fast, small-size and highly power-efficient.

Analysis shows that the signal-to-noise ratio (SNR) of natural neurons is about 100, whereas in the case of analogue VLSI systems it could be better than 10^6. The speed of propagation of signals between neurons of NNN can be estimated as 10 Hz whereas in the case of the analogue ANN it could be several hundred MHz.

This is the ideological basis of the Neuromorphic Systems, interdisciplinary area of science and technology, which is situated between Computational Neuroscience, Physics and Neurobiology. So I would suggest defining *Neuromorphic Systems* as *artificial computational systems, utilising physical phenomena to perform arithmetic (nonbinary) operations*.

It should be noted however that natural brain seems to have a huge advantage in size in comparison with up-to-date VLSI technology (even if it is neuromorphic VLSI). Firstly, the sizes of neural synapses could be very small (tens of nanometers), a great challenge for semiconductor VLSI technology. Secondly, the NNN is a three-dimensional structure, therefore the interconnection problem for such a highly parallel structure (where an output of one neuron could be connected with 1000 to 10000 inputs of other neurons) is solved very efficiently. The problem of 3D neuromorphic systems will be mentioned later in this Introduction.

1.2.2 Drawbacks

In spite of great advantages of analogue technology for ANN implementation, most of ANN hardware is digital. The reason for this is that it is a very nontrivial problem to utilise the potential of analogue technology. For example, the Backpropagation (BP) algorithm (the most popular algorithm of Multilayer Perceptron ANN (MLP) learning) can not be directly implemented on an analogue elements basis because it demands very high precision of the learning circuits. Unlike digital systems (which can be as precise as necessary) analogue systems have restricted precision. There are several parameters of analogue systems contributing to the imprecision: noise, off-sets (mostly caused by inequality of transistors or other elements of the analogue system), nonlinear distortions, time delay (phase distortions), etc.

Analysis of the influence of different kinds of nonidealities is a complex problem and depends on a particular case (a particular neural architecture and a task to be solved). In subsection 3.7 will be described the influence of different kinds of nonidealities of analogue technology on the characteristics of MLP ANN with parallel implementation of the BP algorithm of learning. The different kind of nonidealities affect different parts of ANN (feedforward network and the learning system) in different ways. It will be shown that the feedforward network is sensitive to the noise of the system, that is, accumulating the noise of element of the system) and very tolerant to off-sets and nonlinear distortions. On the other hand the learning system of such parallel-learning ANN is very sensitive to off-sets and to nonlinear distortions but is quite tolerant to the noise of elements of the systems (since the learning of such ANN typically many or-

ders of magnitude slower than the signal propagation through the feedforward network of the ANN, therefore noise is averaged in the learning system).

1.2.3 Prospects (analogue technology as a necessary attribute of the future high-complexity computational systems)

To suppress the harmful influence of analogue element off-sets on the learning system, a special system of "self-correction" of high precision circuits of analogue ANN is necessary. Several kinds of such "self-correction" system for parallel-learning analogue MLP ANN are described in subsection 3.7.

The purpose of this self-correction system is the creation of a high precision system on the basis of imprecise elements. To achieve this aim a special phase of the ANN function (the self-correction phase) is necessary. So, such analogue parallel-learning ANN have three phases of function:

- the "working" phase, when the nonlinear transform is fixed (all weights are constant) and the aim of ANN is nonlinear processing of input information;
- the learning phase, when the ANN must adjust its weights aiming to minimise the error of ANN or maximise the parameter of quality of the ANN;
- the self-correction phase, when the self-correction system is adjusting the high precision units of the learning system (zeroed error signal, fixed weights).

In ideal (or digital) ANN there are only the two first above mentioned phases.

Such a 3-phase scheme of function is typical for natural neural networks (which are based on **analogue** mechanisms of nonlinear transform, and therefore have off-sets, which must be suppressed). *In this case the self-correction stage may be taking place during the sleep* of an animal or a human. Thus the structure of analogue ANN is, generally, more complex than the structure of the digital ANN (see Fig. 1). At the same time the complexity of self-correction systems is normally less than the complexity of the feedforward network or learning system (and the time which it takes for the self-correction procedure is less than the time for learning). **Therefore the gains in power consumption and size overwhelmingly prevails over the necessary increased complication of systems based on analogue elements.**

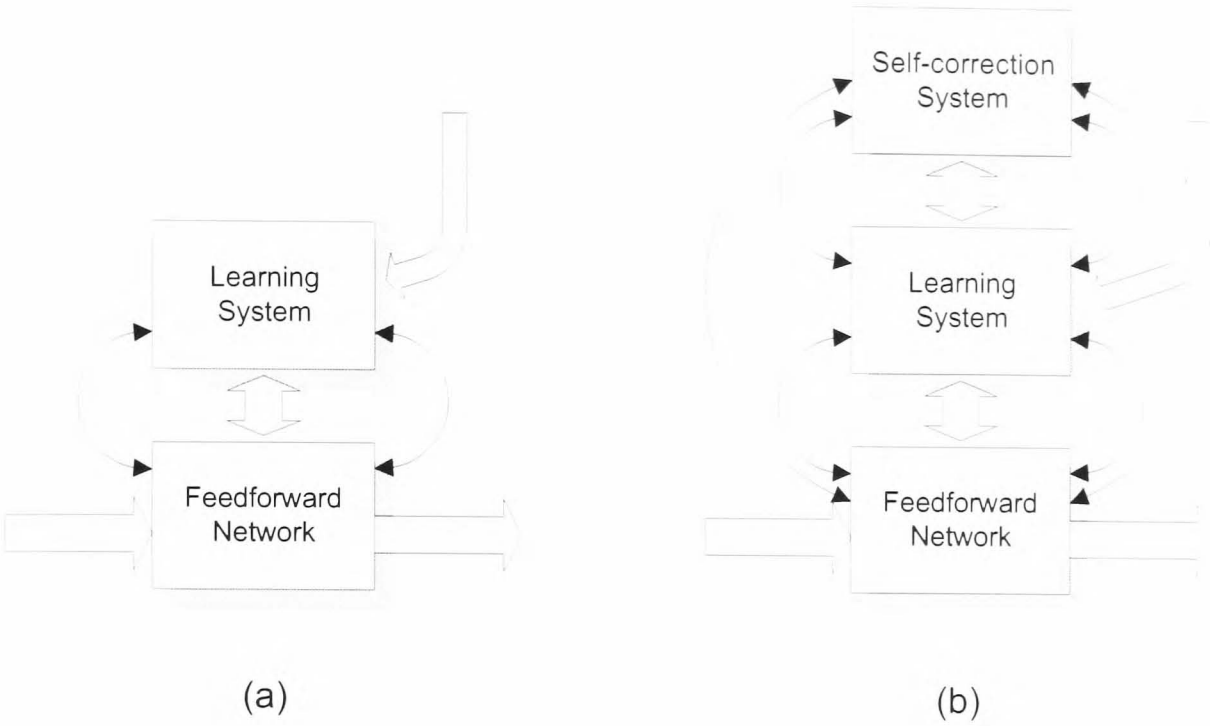


Figure 1. The block-schemes of (a) ideal (digital) neural network; (b) analogue parallel-learning neural network

The important point is that some of the self-correction systems of analogue ANN have a property not only to suppress small off-sets of high precision learning circuits, but also are capable of localising the harmful influence of broken element(s) of the ANN (irrespective of whether it is an element of the feedforward network, the learning system or of the self-correction system). This problem will be discussed in subsection 3.7. This property of some analogue architectures could be very important for creation of superreliable systems, as well as in cases of not very reliable (with relatively high probability of defects) Ultra Large Scale Integration (ULSI) technologies for creation of **analogue** ANNs. It is an important advantage of analogue technology since the breakage (or defect) of only one transistor causes the crash of the whole **digital** processor.

The potential property of analogue ANNs of high tolerance to breakage or defects of their elements (along with the advantage in power consumption) could be very important e.g. for creation of multi-layer integrated structures. In this case the vitality of analogue systems is very important since in the case of ultra-complex (especially three-dimensional) systems some defects will inevitably occur somewhere in the structure. The harm from these defects should be minimised by a proper hardware architecture.

To justify the prospects and the reality of multilayer integrated technology for analogue ANN implementation, let us consider a quasi-3D integrated structure which could be composed on the basis of up-to-date VLSI technology.

Let us suggest that a wafer-sized analogue VLSI implementation of ANN was created. As a next step, let us suggest that 100 such wafers were pasted together, so that overall thickness (and other dimensions) of the structure is less than 10 cm. Let us estimate the characteristics of the resulting structure. We will assume that the analogue VLSI system performing the same quantity of calculations as a digital one could be 10 times smaller in sizes (thus 100 times less area) and take 10^5 times less power. Note that the area of a Pentium III crystal is about 1 cm^2 whereas the power consumption is about 30 Watt. The above sandwiched structure will be just 10 times more power consuming than the Pentium III (500MHz) processor and 10^5 times more computationally powerful. Such a structure will be almost as computationally powerful as a human brain (see the above estimations of the human brain's computational power, which is about 10^6 times faster than the Pentium).

The fact that such a naive structure (which can not be called a real 3D structure since there are no interconnections between layers) could be so computationally powerful is quite surprising. Nevertheless it doesn't mean that we are close to creation of artificial brain.

Firstly, the great question is what is the computational architecture of AI systems? For several decades mathematicians have been working on this problem. It is obvious however that it is impossible to succeed in this problem without experiments. Computational experiments based on computer simulations seem unpromising because of far too low computational power. In this connection the creation of computationally powerful systems will be vital for progress in establishing of AI architecture .

The second reason why the creation of AI systems is a great challenge for technology is the huge (~10^5 Gbyte) memory of natural brain. It is very fundamental problem since the sizes of synapses of the natural brain could be just tens of nanometers. Since the ANN for artificial brain are supposed to be very interconnected, the technological problem of local massive interconnection is also of great importance. To solve this problem it would not be enough just to develop existing VLSI technology. Probably some principally different approach is necessary. Such systems definitely must be three

dimensional (i.e. multilayer integrated structures) In such super-complex systems, the tolerance to element degradation will be very important.

Conclusions

1. The computational power of artificial systems could become comparable with that of the human brain in the foreseeable future. The creation of systems of artificial intelligence however is a much greater problem since firstly the proper computational architecture has to be established and secondly, the problem of huge ($\sim 10^5$ Gbyte) memory of the AI system must be solved in hardware.

2. A special self-correction system is necessary for creation of highly efficient analogue neuromorphic systems.

3. The low power consumption of analogue ANN gives the opportunity of creating the multilayer integrated structure, thus of increasing the scale of integration by several (more than four) orders of magnitude.

1.3 Other paradigms of highly efficient computational systems

The creation of artificial brain is definitely the greatest aim for researchers and developers of artificial neural networks. This goal will not be reached in the next few decades. On the other hand some of the simplest neural systems could be created and find commercial applications in the near future. These systems will be able to solve tasks of image/speech processing/recognition, multiparameter optimisation, etc.

Such systems will not be as universal as the computer or as a human brain or AI systems, but could be very efficient since they could be optimised for particular problems, will be based on principles of rationality and will utilise the advantages of analogue adaptive technology (high speed, small sizes and low power consumption).

Let us consider the two principles which are widely used by signal processing systems but seems to be out of use by the natural information processing systems: polynomial representation and the principle of orthogonality.

Polynomial transforms and approximation. As was already mentioned, neuromorphic systems should not mimic natural systems, but must use nonlinear physical phenomena as computational primitives and should be built on the basis of principles of rationality.

One of the main tasks which neural networks (both natural and artificial) are solving is the task of approximation.

Natural neural networks seem to be mostly using the universal technique of approximation based on sigmoid-like functions $f(x) = sigm(x)$ (e.g. $f(x) = \frac{1}{1+\exp(-x)}$), which, in accordance with the Kolmogorov's theorem about approximation (see [10]), indeed are capable of approximation of any function of any number of arguments $\Phi(x_1, x_2, x_3, ...x_N)$. The sigmoid has a linear part when the argument x is near zero: $-1 < x < 1$ and is constant (saturated) if $x \ll -1$ or $1 \ll x$ (see Fig. 2).

At the same time, it is known from the theory of approximation that an optimal approximation of *different classes of functions* demands *different basic functions*. "Optimal" means that the minimal quantity of degrees of freedom are to be adjusted to get the required precision of approximation. The functions to be approximated could be finite or they could be periodical. They could have different degree of smoothness. The precision of approximation could also be different in different cases. For each problem

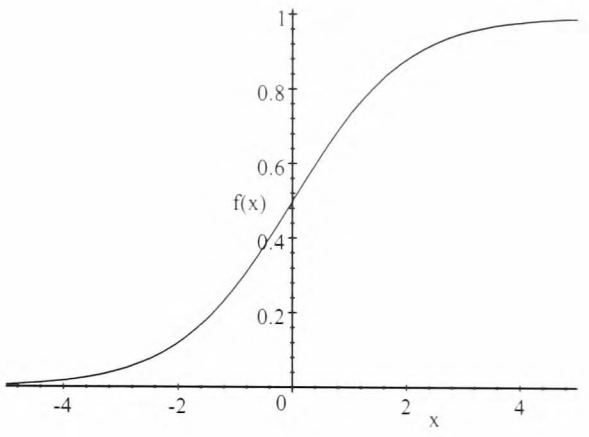


Figure 2. An example of sigmoid function

an appropriate basis of approximating functions must be established and used. Therefore, ANN dedicated to solve particular problems must have proper (relevant to the problem) activation functions.

In real life the tasks (and subtasks) to be solved vary in a very wide range. Therefore the ANN should be built on the basis of *a priori* knowledge about the task to be solved, otherwise the efficiency of ANN will dramatically decrease. For example, the human brain become very inefficient solving the problems of high precision calculations or complex logical tasks, which could be easily solved by a simplest calculator. So the ANN should be **task-specific** rather than **universal** systems.

One particular case is very important. A very wide class of physical phenomena and analogue systems are weakly nonlinear. That's why the polynomial representation and approximation for modelling of such phenomena are widespread in science and technology.

The efficiency (simplicity) and high accuracy of polynomial approximation of weakly nonlinear functions is the main motivation of polynomial-based ANN. The most well-known polynomial ANN are the so-called Volterra neural networks (see e.g. [11]). In the section 3.1 several examples of polynomial-based analogue adaptive systems are described.

Another useful application of polynomials is a result of the fact that many very important orthogonal bases can be represented by polynomials. For example: the Chebyshev polynomials $T_n(x)$ give us sine and cosine bases of functions: $T_n(\sin(t)) = \sin(nt)$.

The Legendre, Laguerre and Hermit polynomials as well as some other functions also play very important roles in signal processing, telecommunications, etc. In section 3.2 the analogue synthesiser of different orthogonal functions is described.

Several examples of the application of polynomial-based analogue systems are described in sections 3.3 and 3.6

The principle of orthogonality. Orthogonal functions play a very important role in mathematics, physics, informatics, image/signal processing, telecommunication and other branches of science and technology. The most famous orthogonal functions nowadays are sine/cosine functions which make up the Fourier basis. For several years, the wavelet family of functions (specific finite orthogonal functions) became very important in image/signal processing.

Orthogonal functions are so important since (as it is known from mathematical analysis) an arbitrary piecewise continuous function $\Phi(x)$ could be efficiently represented as an expansion in terms of orthonormal (orthogonal normalised) functions $f_i(x)$: $\Phi(x) \simeq \sum c_i f_i(x)$,where $c_i = \int \Phi(x) f_i(x) dx$ (see [42]).

If the f_i basis of functions was not orthogonal, the expansion of $\Phi(x)$ in terms of f_i functions (if f_i are *complete*): $\Phi(x) \simeq \sum c_i f_i(x)$ still will be possible but the calculation of c_i coefficients will be much more complex.

The selection of appropriate orthogonal functions $\Phi_i(x)$ allows highly efficient characterisation of a function $f(x)$. For example: the fact that the first terms of the Fourier basis are smoother than the higher order terms allows one to obtain the smooth approximation of the function $f(x)$ by a truncated basis $\Phi_i(x)$ where $i < N$. The selection of N allows us to get the desirable smoothness of the approximating function. Thus if we have an *a priori* knowledge that the function $\Phi_i(x)$ is a smooth function with added noise, the approximation by a truncated Fourier basis allows elimination of noise.

We can mention several other examples when *a priori* knowledge about functions could be utilised by means of selection of appropriate basis of functions.

Thus the bigger variety of orthogonal bases the greater the chance to realise in full the *a piori* knowledge about the system to be modelled or characterised.

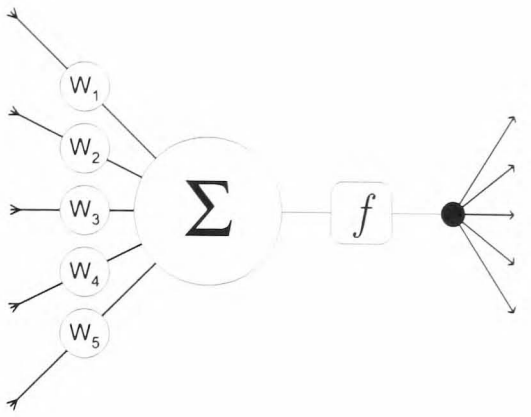


Figure 3. Typical scheme of neuron, constituting the MLP ANN

This fact was the main motivation for the creation of the Analogue Synthesiser of Orthogonal Functions, described in section 3.2, which let us to synthesise a wide variety of orthogonal function.

Another stimulus was the fact that analogue neuromorphic systems could be much more efficient if they utilise the principle of orthogonality. Actually the Synthesiser is a specific analogue adaptive system, which could be used as a unit to build a more complex purely analogue information processing system.

Orthogonal functions are already used by some ANN architectures. For example, the Chebyshev polynomials-based (CPB) unified model neural network is described in [12]. The fast training polynomial approximator based on Legendre polynomials is described in [13].

The remarkable properties of the above mentioned approximators are their very fast adaptation as well as the guaranteed convergence to the **global** minimum of error. In section 3.1 there is a comparison of the power-type functions-based polynomial approximator with the Legendre polynomials-based approximator. This comparison demonstrates the importance of the principle of orthogonality.

The advantages of the orthogonal transforms also become apparent when one compare the analogue homomorphic video-filter (which is based on orthogonal transforms and described in section 3.5) with the Silicon Retina (which is based on nonorthogonal parallel image processing and attempts to mimic the natural retina, see [6]).

Orthogonal and polynomial systems* vs. *multi-layer perceptron In spite of the importance of the principle of orthogonality and the polynomial representation, the

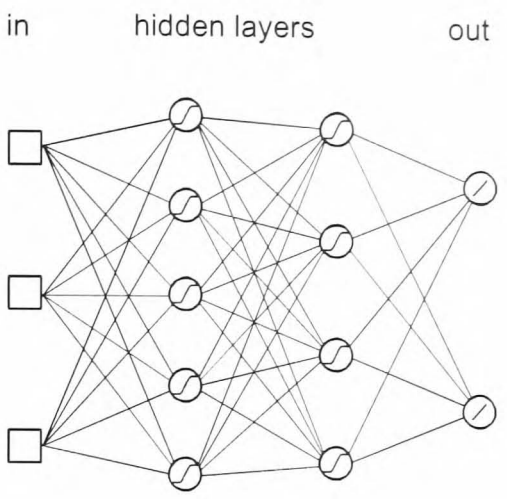


Figure 4. An example of MLP neural network architecture

Multi-Layer Perceptron ANN architecture (see Fig. 3 and Fig. 4) based e.g. on the sigmoid activation function, seems to me to be the basis for implementation of AI systems.

The reasons are:

1. the sigmoid-based approximators (ANN) cover a very wide range of functions to be approximated. Indeed, in the case of large weights, the ANN will behave similar to the digital scheme, whereas in the case of weak weights the neural network will have weakly-nonlinear transfer functions (thus will allow approximation of smooth interrelations). The polynomial-based ANN are efficient only in case of weak nonlinearity of functions to be approximated, and in cases of stronger nonlinearity (higher than 6th degree of polynomial) their use becomes inefficient.

2. The advantage of orthogonal approximators of fast adaptation to the global minimum of error does not work in cases of standard ANN tasks. This is because normally the data set to be processed by ANN is not regular (but normally is some sparse irregular multi-dimensional set of data) and in such cases the orthogonal functions (which assumes orthogonality over some interval) will lose their orthogonality, i.e. $\sum_k f_i(\overrightarrow{x_k}) \cdot f_j(\overrightarrow{x_k}) \neq \delta_{ij}$, where δ_{ij}.is the Kronecker delta: $\delta_{ij} = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j \end{cases}$, k is the number of data samples used during ANN learning.

Therefore the slowness of learning of the sigmoid-based MLP ANN seems like the payment for its potentially extremely high flexibility, and its efficiency of use of its degrees of freedom in the cases of substantially nonlinear functions to be approximated/interpolated.

One of the promising directions of research in my view will be aiming to utilise the principle of orthogonality in a specific ANN, dedicated to solve standard ANN tasks. Such ANN could contain some adaptive units, which will **locally** orthogonalise and normalise the data propagating through the ANN. I hope such an approach will let us increase both the speed of learning and the probability of convergency of the ANN to the global minimum of error, and will be able to solve the same spectrum of problems as the MLP ANN.

Conclusions

1. Polynomial-based adaptive analogue systems as well as adaptive systems utilising the principle of orthogonality, could be very efficient and could find commercial applications already in the near future.

2. In many typical ANN tasks (especially in cases of complex tasks) multi-layer perceptron neural architecture could be much more efficient than polynomial-based adaptive systems. In many cases the principle of orthogonality is not applicable.

Chapter 2
Review (relevant biological, mathematical and technological background and up-to-date results and developments)

2.1 Basics of neural networks

2.1.1 Biological neurons

A human brain consists of approximately 10^{11} computing elements called neurons. They communicate through a connection network of axons and synapses having a density of approximately 10^4 synapses per neuron. Our hypothesis regarding the modeling of the natural nervous system is that neurons communicate with each other by means of electrical impulses ([14]). The neurons operate in a chemical environment that is even more important in terms of actual brain behavior. We thus can consider the brain to be a densely connected electrical switching network conditioned largely by the biochemical processes. The vast neural network has an elaborate structure with very complex interconnections. The input to the network is provided by sensory receptors. Receptors deliver stimuli both from within the body, as well as from sense organs when the stimuli originate in the external world. The stimuli are in the form of electrical impulses that convey the information into the network of neurons. As a result of information processing in the central nervous systems, the effectors are controlled and give human responses in the form of diverse actions. We thus have a three-stage system, consisting of receptors, neural network, and effectors, in control of the organism and its actions.

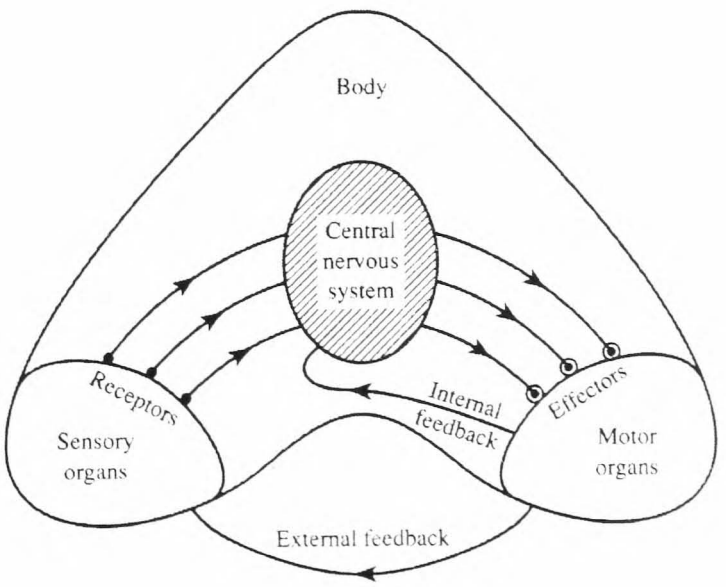


Figure 5. Information flow in nervous system.

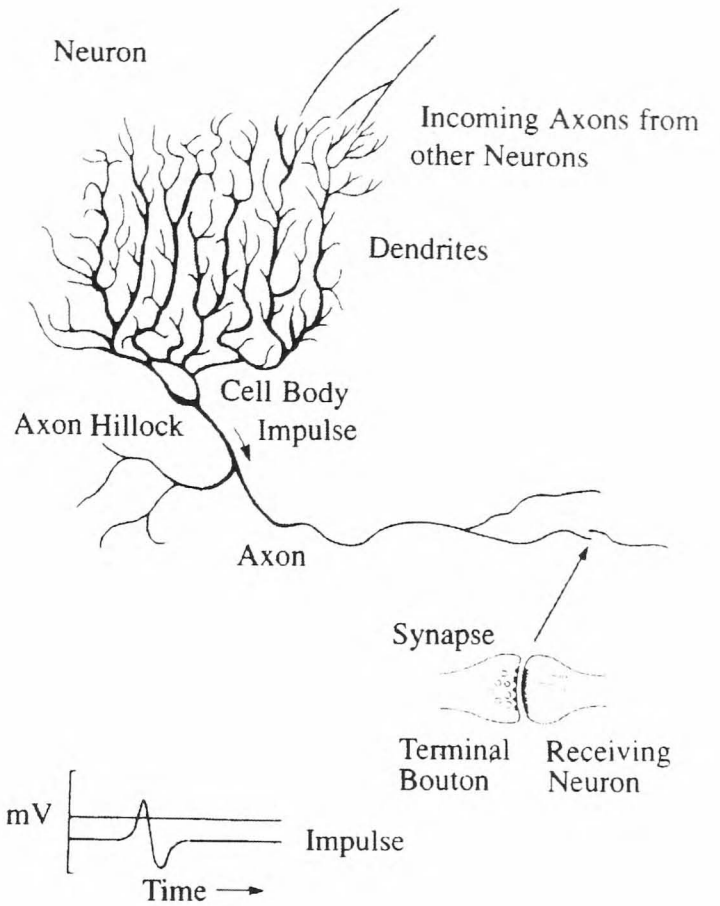


Figure 6. Schematic diagram of a neuron and a sample of pulse train. ([1])

A lucid, although rather approximate idea, about the information links in the nervous system is shown in Figure 5. As we can see from the figure, the information is processed, evaluated, and compared with the stored information in the central nervous system. When necessary, commands are generated there and transmitted to the motor organs. Notice that motor organs are monitored in the central nervous system by feedback links that verify their action. Both internal and external feedback control the implementation of commands. As can be seen, the overall nervous system structure has many of the characteristics of a closed-loop control system.

The elementary nerve cell, called a neuron, is the fundamental building block of the biological neural network. Its schematic diagram is shown in Figure 6.

A typical cell has three major regions: the cell body, which is also called the soma, the axon, and the dendrites. Dendrites form a dendritic tree, which is a very fine bush

of thin fibers around the neurons body. Dendrites receive information from neurons through axons – long fibers that serve as transmission lines. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits into a fine arborization. Each branch of it terminates in a small endbulb almost touching the dendrites of neighboring neurons. The axon-dendrite contact organ is called a synapse. The synapse is where the neuron introduces its signal to the neighboring neuron. The signals reaching a synapse and received by dendrites are electrical impulses. The interneuronal transmission is sometimes electrical but is usually effected by the release of chemical transmitters at the synapse. Thus, terminal-boutons generate the chemical that affects the receiving neuron. The receiving neuron either generates an impulse to its axon, or produces no response. The neuron is able to respond to the total of its inputs aggregated within a short time interval called the period of latent summation. The neuron's response is generated if the total potential of its membrane reaches a certain level. The membrane can be considered as a shell, which aggregates the magnitude of the incoming signals over some duration. Specifically, the neuron generates a pulse response and sends it to its axon only if the conditions necessary for firing are fulfilled.

Let us consider the conditions necessary for the firing of a neuron. Incoming impulses can be excitatory if they cause the firing, or inhibitory if they hinder the firing of the response. A more precise condition for firing is that the excitation should exceed the inhibition by the amount called the threshold of the neuron, typically a value of about -40mV ([14]). Since a synaptic connection causes the excitatory or inhibitory reactions of the receiving neuron, it is practical to assign positive and negative unity weight values, respectively, to such connections. This allows us to reformulate the neuron's firing condition. The neuron fires when the total of the weights to receive impulses exceeds the threshold value during the latent summation period.

The incoming impulses to a neuron can only be generated by neighboring neurons and by the neuron itself. Usually, a certain number of incoming impulses are required to make a target cell fire. Impulses that are closely spaced in time and arrive synchronously are more likely to cause the neuron to fire. As mentioned before, observations have been made that biological networks perform temporal integration and summation of incoming signals. The resulting spatio-temporal processing performed by natural neural networks is a complex process and much less structured than digital computation. The neural impulses are not synchronized in time as opposed to the synchronous discipline

of digital computation. The characteristic feature of the biological neuron is that the signals generated do not differ significantly in magnitude; the signal in the nerve fiber is either absent or has the maximum value. In other words, information is transmitted between the nerve cells by means of binary signals.

After carrying a pulse, an axon fiber is in a state of complete nonexcitability for a certain time called the refractory period. For this time interval the nerve does not conduct any signals, regardless of the intensity of excitation. Thus, we may divide the time scale into consecutive intervals, each equal to the length of the refractory period. This will enable a discrete-time description of the neuron's performance in terms of their states at discrete time instances. For example, we can specify which neurons will fire at the instant $k+1$ based on the excitation conditions at the instant k.The neuron will be excited at the present instant if the number of excited excitatory synapses exceeds the number of excited inhibitory synapses at the previous instant by at least the number T, where T is the neuron's threshold value.

The time units for modeling biological neurons can be taken to be of the order of a millisecond. However, the refractory period is not uniform over the cells. Also, there are different types of neurons and different ways in which they connect. Thus, the picture of real phenomena in the biological neural network becomes even more involved. We are dealing with a dense network of interconnected neurons that release asynchronous signals. The signals are then fed forward to other neurons within the spatial neighborhood but also fed back to the generating neurons.

The above discussion is extremely simplified when seen from a neurobiological point of view, though it is valuable for gaining insight into the principles of biological computation. Our computing networks are far simpler than their biological counterparts. Let us examine an artificial neuron model that is of special, historical significance.

2.1.2 McCulloch-Pitts Neuron Model

The first formal definition of a synthetic neuron model based on the highly simplified considerations of the biological model described in the preceding section was formulated by McCulloch and Pitts [15]. The McCulloch-Pitts model of the neuron is shown in Figure 7a. The inputs x_i, for $i = 1, 2, ..., N$, are 0 or 1, depending on the

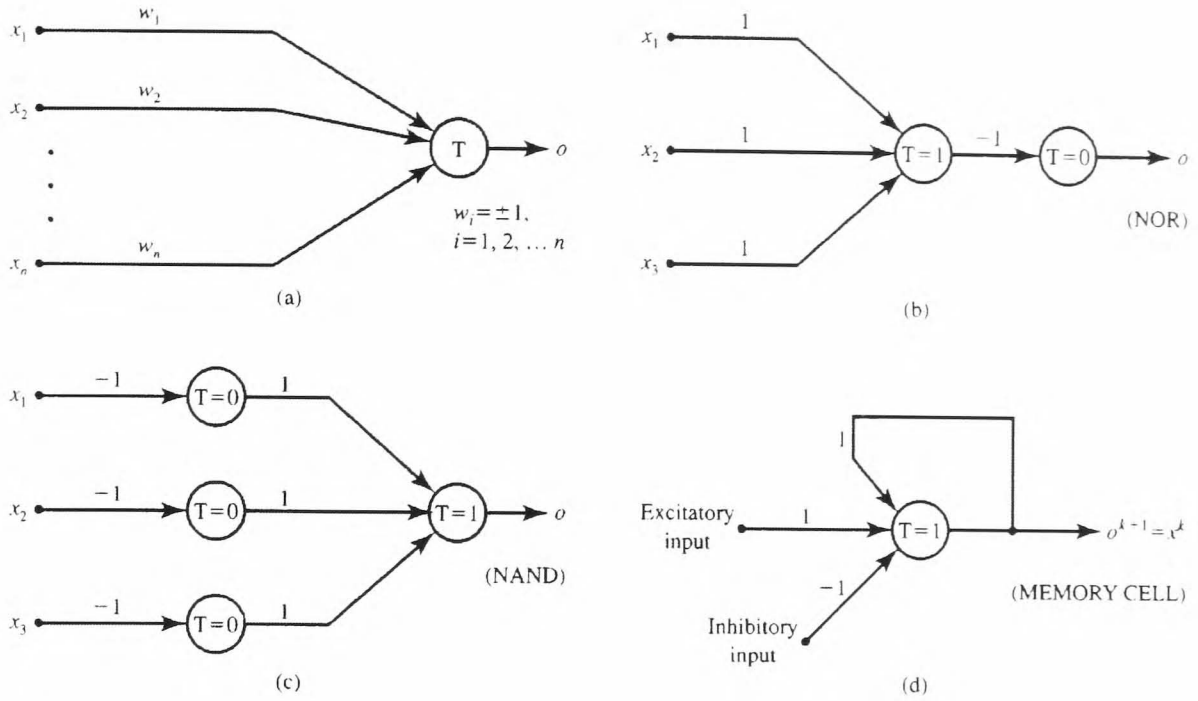


Figure 7. McCulloch-Pitts model neuron and elementary logic networks: (a) model diagram, (b) NOR gate, (c) NAND gate and (d) memory cell.

absence or presence of the input impulse at instant k. The neuron's output signal is denoted as o. The firing rule for this model is defined as follows

$$o^{k+1} = \begin{cases} 1 \text{ if } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 \text{ if } \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$

where superscript k = 0, 1, 2, ... denotes the discrete-time instant, and w_i is the multiplicative weight connecting the i'th input with the neuron's membrane. In further discussion, we will assume that a unity delay elapses between the instants k and $k+1$. Note that $w_i = +1$ for excitatory synapses, $w_i = -1$ for inhibitory synapses for this model, and T is the neuron's threshold value, which needs to be exceeded by the weighted sum of signals for the neuron to fire.

Although this neuron model is very simplistic, it has substantial computing potential. It can perform the basic logic operations NOT, OR, and AND, provided its weights and thresholds are appropriately selected. As we know, any multivariable combinational function can be implemented using either the NOT and OR, or alternatively the NOT and AND, Boolean operations. Examples of three-input NOR and NAND gates using the McCulloch-Pitts neuron model are shown in Figure 7(b) and (c).

Both the neuron model and the example logic circuits discussed so far have been combinational and little attention has been paid to the inherent delay involved in their

operation. However, the unity delay property of the McCulloch–Pius neuron model maker it possible to build sequential digital circuitry.

First note that a single neuron with a single input x and with the weight and threshold values both of unity, computes $o^{k+1} = x^k$. Such a simple network thus behaves as a single register cell able to retain the input for one period elapsing between two instants. As a consequence once a feedback loop is closed around the neuron as shown in Figure 7(d). we obtain a memory cell. An excitatory input of 1 initializes the firing in this memory cell, and an inhibitory input of 1 initializes a nonfiring state. The output value, at the absence of inputs, is then sustained indefinitely. This is because the output of 0 fed back to the input does not cause firing at the next instant, while the output of 1 does.

Thus, we see that digital computer hardware of arbitrary complexity can be constructed using an artificial neural network consisting of elementary building blocks as shown in Figure 7. Our purpose, however, is not to duplicate the function of already efficient digital circuitry, but rather to assess and exploit the computational power that is manifested by interconnected neurons subject to the experiential learning process.

2.1.3 Neuron Modeling for Artificial Neural Systems

The McCulloch-Pitts model of a neuron is characterized by its formalism and its elegant, precise mathematical definition. However, the model makes use of several drastic simplifications. It allows binary 0, 1 states only, operates under a discrete-time assumption, and assumes synchrony of operation of all neurons in a larger network. Weights and the neurons' thresholds are fixed in the model and no interaction among network neurons takes place except for signal flow. Thus, we will consider this model as a starting point for our neuron modeling discussion. Specifically, the artificial neural systems and computing algorithms employ a variety of neuron models that have more diversified features than the model just presented. Below, we introduce the main artificial neuron models.

Every neuron model consists of a processing element with synaptic input connections and a single output. The signal flow of neuron inputs, x_i, is considered to be unidirectional as indicated by arrows, as is a neuron's output signal flow. A general neuron symbol is shown in Figure 8. This symbolic representation shows a set of weights and the neurons processing unit or node. The neuron output signal is given by

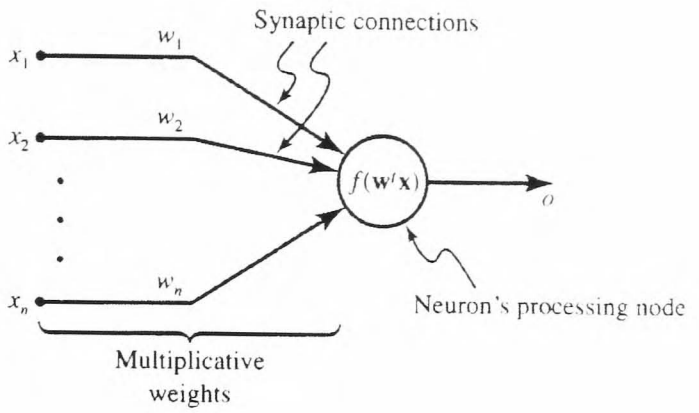


Figure 8. General symbol of neuron consisting of processing node and synaptic connections

the following relationship:

$$o = f(\mathbf{w}^T\mathbf{x}), \text{ or } o = f(\sum_{i=1}^{n} w_i x_i)$$

where $\mathbf{w}$ is the weight vector defined as

$$\mathbf{w} \equiv [w_1 w_2 ... w_n]^T$$

and $\mathbf{x}$ is the input vector:

$$\mathbf{x} \equiv [x_1 x_2 ... x_n]^T$$

(All vectors defined in this text are column vectors; superscript T denotes a transposition.) The function $f(\mathbf{w}^T\mathbf{x})$ is often referred to as an activation function. Its domain is the set of activation values, net, of the neuron model, we thus often use this function as $f(net)$. The variable net is defined as a scalar product of the weight and input vector

$$net \equiv \mathbf{w}^T\mathbf{x}$$

The argument of the activation function, the variable net, is an analogue of the biological neuron's membrane potential. Note that the threshold value is not explicitly used in the above equations, but this is only for notational convenience. We have momentarily assumed that the modeled neuron has $n-1$ actual synaptic connections that come from actual variable inputs $x_1, x_2, ..., x_{n-1}$. We have also assumed that $x_n = 1$ and $w_n = -T$. Since threshold plays an important role for some models, we will sometimes need to extract explicitly the threshold as a separate neuron model parameter.

After the operation of summation of its weighted inputs the neuron performs the nonlinear operation $f(net)$ through its activation function. Typical activation functions

used are

$$f(net) \equiv \tanh(\lambda \cdot net) \tag{1}$$

and

$$f(net) \equiv sgn(net) = \begin{cases} +1, \ net \geq 0 \\ -1, \ net < 0 \end{cases} \tag{2}$$

where $\lambda > 0$ is proportional to the neuron gain determining the steepness of the continuous function $f(net)$ near $net = 0$. The continuous activation function is shown in Figure 9(a) for various λ. Notice that as $\lambda \rightarrow \infty$, the limit of the continuous function becomes the $sgn(net)$ function. Activation functions (2.1) and (2.2) are called bipolar continuous and bipolar binary functions, respectively. The word bipolar is used to point out that both positive and negative responses of neurons are produced for this definition of the activation function.

By shifting and scaling the bipolar activation functions defined by (2.1) and (2.2), unipolar continuous and unipolar binary activation functions can be obtained, respectively, as

$$f(net) \equiv \frac{1}{1 + exp(-\lambda \cdot net)}$$

and

$$f(net) \equiv \begin{cases} +1, \ net \geq 0 \\ 0, \ net < 0 \end{cases}$$

(see Figure 9(b).) Again, the unipolar binary function is the limit of continuous $f(net)$ when $\lambda \rightarrow \infty$. The soft-limiting activation functions are often called sigmoidal char-

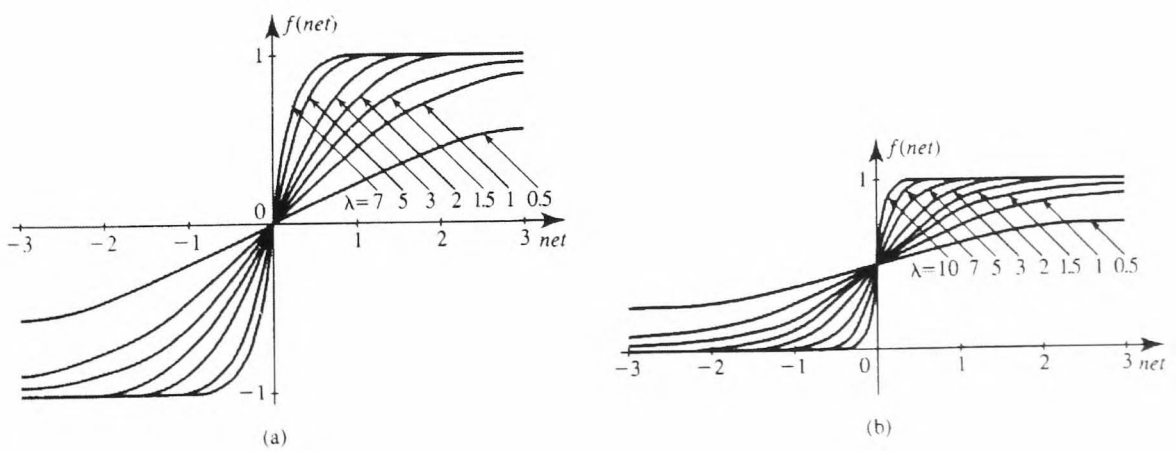


Figure 9. Activation functions of a neuron: (a) bipolar continuous and (b) unipolar continuous.

acteristics, as opposed to the hard-limiting activation functions given. Hard-limiting activation functions describe the discrete neuron model. Most neurons employ bipolar activation functions. Some neural network architectures or applications do, however, specifically require the unipolar neuron responses. If this is the case, appropriate qualification for the type of activation function used is made.

Some neural models require the use of another type of nonlinearity than that defined in the above equations.

Artificial neural systems using the models defined by the above equations do not involve the biological neuron features of delay, refractory period, or discrete-time operation. In fact, the neuron models listed so far in this section represent instantaneous, memoryless networks; i.e. they generate the output response determined only by the present excitation. A delay feature can be added to the instantaneous neuron model by adding an external delay element to make the ensemble of neurons operate with memory.

2.1.4 Learning and adaptation

Each of us acquires and then improves our skills and abilities through the basic phenomenon of learning. Learning is a fundamental subject for psychologists. In general, learning is a relatively permanent change in behavior brought about by experience. Learning in human beings and animals is an inferred process; we cannot see it happening directly and we can assume that it has occurred by observing changes in performance. Learning in neural networks is a more direct process, and we typically can capture each learning step in a distinct cause-effect relationship. To perform any of the processing tasks discussed in the previous section, neural network learning of an input-output mapping from a set of examples is needed. Designing an associator or a classifier can be based on learning a relationship that transforms inputs into outputs given a set of examples of input-output pairs.

Our focus in this section will be artificial neural network learning rules. A neuron is considered to be an adaptive element. Its weights are modifiable depending on the input signal it receives, its output value, and the associated teacher response. In some cases the teacher signal is not available and no error information can be used, thus the

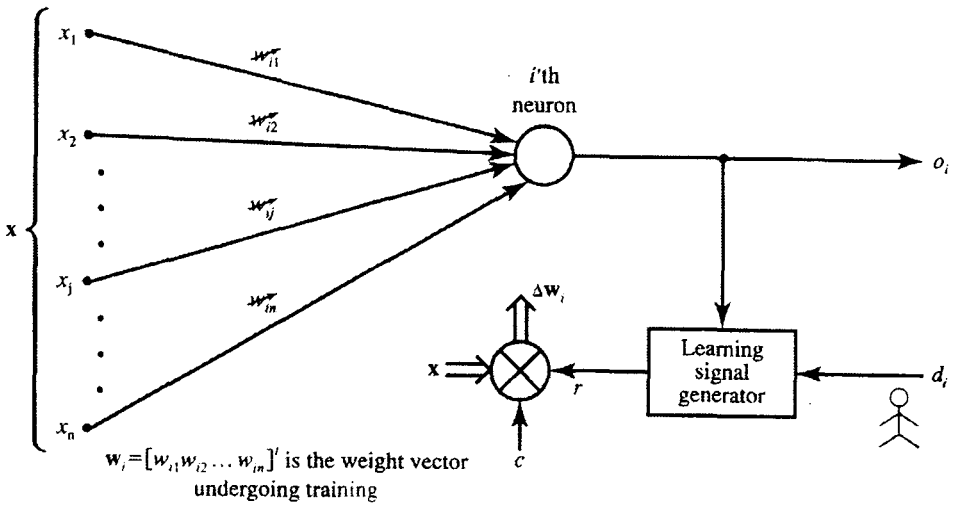


Figure 10. Illustration for weight learning rules (d_i provided only for mode).

neuron will modify its weights based only on the input and / or output. This is the case for unsupervised learning.

Let us study the learning of the weight vector $\mathbf{w}_i$, or its components w_{ij} connecting the j'th input with the i'th neuron. The trained network is shown in Figure 10 and uses the neuron symbol from Figure 8. In general, the j'th input can be an output of another neuron or it can be an external input. Our discussion in this section will cover single-neuron and single-layer network supervised learning. Under different learning rules, the form of the neurons activation function may be different. Note that the threshold parameter may be included in learning as one of the weights. This would require fixing one of the inputs, say x_n. We will assume here that x_n, if fixed, takes the value of -1.

The following general learning rule is adopted in neural network studies ([16]): The weight vector $w1 = [w_{i1} \quad w_{i2} \quad \ldots \quad w_{in}]^T$ increases in proportion to the product of input $\mathbf{x}$ and learning signal δ. The learning signal δ is in general a function of $\mathbf{w}_i, \mathbf{x}$, and sometimes of the teacher's signal d_i. We thus have for the network shown in Figure 10:

$$\delta = \delta(\mathbf{w}_i, \mathbf{x}, d_i) \tag{3}$$

The increment of the weight vector $\mathbf{w}_i$ produced by the learning step at time t according to the general learning rule is

$$\Delta\mathbf{w}_i(t) = \alpha\delta[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)]\mathbf{x}(t) \tag{4}$$

where α is a positive number called the learning constant that determines the rate of learning. The weight vector adapted at time t becomes at the next instant, or learning step,

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha\delta[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)]\mathbf{x}(t) \tag{5}$$

The superscript convention will be used in this text to index the discrete-time training steps as in Eq.(2.5). For the k'th step we thus have from (2.5) using this convention

$$\mathbf{w}_i^k = \mathbf{w}_i^k + \alpha\delta[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)]\mathbf{x}(t) \tag{6}$$

The learning in 2.6 assumes the form of a sequence of discrete-time weight modifications. Continuous-time learning can be expressed as

$$\frac{d\mathbf{w}_i(t)}{dt} = \alpha\delta[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)]\mathbf{x}(t)$$

Discrete-time, or stepwise, learning is reviewed below. Weights are assumed have been suitably initialized before each learning experiment started.

2.1.5 Delta Learning Rule

The delta learning rule is only valid for continuous activation functions as defined e.g. in 2.1, and in the supervised training mode. The learning signal for this rule is called $delta$ and is defined as follows

$$\delta \equiv [d_i - f(\mathbf{w}_i^T\mathbf{x})]f'(\mathbf{w}_i^T\mathbf{x}) \tag{7}$$

The term $f'(\mathbf{w}_i^T\mathbf{x})$ is the derivative of the activation function $f(net)$ computed for $net = \mathbf{w}_i^T\mathbf{x}$. The explanation of the delta learning rule is shown in Figure 11. This learning rule can be readily derived from the condition of least squared error between o_i and d_i. Calculating the gradient vector with respect to $\mathbf{w}_i$ of the squared error defined as

$$E \equiv \frac{1}{2}(d_i - o_i)^2$$

which is equivalent to

$$E \equiv \frac{1}{2}(d_i - f(\mathbf{w}_i^T\mathbf{x}))^2$$

we obtain the error gradient vector value

$$\nabla E = -[d_i - o_i]f'(\mathbf{w}_i^T\mathbf{x})\mathbf{x} \tag{8}$$

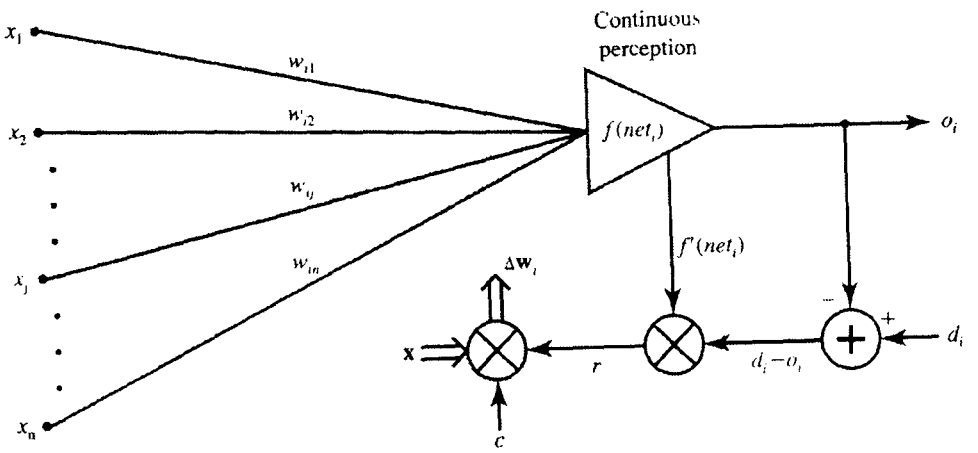


Figure 11. Delta learning rule

Since the minimization of the error requires the weight changes negative gradient direction, we take

$$\Delta \mathbf{w}_i = -\eta \nabla E \tag{9}$$

where η is a positive constant. We then obtain from Eqs. (2.8) and (2.9)

$$\Delta \mathbf{w}_i = \eta[d_i - o_i)] f'(\mathbf{w}_i^T \mathbf{x})\mathbf{x} \tag{10}$$

Note that the weight adjustment as in (2.10) is computed based on minimization of the squared error. Considering the use of the general learning rule (2.4) and plugging in the learning signal as defined in (2.7), the weight adjustment becomes

$$\Delta \mathbf{w}_i = \alpha[d_i - o_i] f'(\mathbf{w}_i^T \mathbf{x})\mathbf{x}. \tag{11}$$

Therefore, it can be seen that (2.10) is identical to (2.11), since α and η have been assumed to be arbitrary constants. The weights are initialized at any values for this method of training.

The delta rule was introduced only recently for neural network training (Rumelhart [17]). This rule parallels the discrete perceptron training rule. It also can be called the continuous perceptron training rule. The delta learning rule can be generalized for multilayer networks.

2.1.6 Multilayer Perceptrons

In this subsection we consider an important class of neural networks, namely, multilayer feedforward networks. Typically, the network consists of a set of sensory units (source nodes) that constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are commonly referred to as multilayer perceptrons (MLPs), which represent a generalization of the single-layer perceptron considered in previous subsections.

Multilayer perceptions have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm. This algorithm is based on the error–correction learning rule. As such, it may be viewed as a generalisation of an equally popular adaptive filtering algorithm: the ubiquitous least-mean-square (LMS)

algorithm (or delta-rule algorithm) described in previous subsection for the special case of a single linear neuron model.

Basically, the error back-propagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through the network, layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the network are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with the error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections hence the name "error back-propagation". The synaptic weights are adjusted so as to make the actual response of the network move closer to the desired response. The error back-propagation algorithm is also referred to in the literature as the back-propagation algorithm, or simply back-prop. The learning process performed with the algorithm is called back-propagation learning.

A multilayer perceptron has three distinctive characteristics:

1. The model of each neuron in the network includes a nonlinearity at the output end. The important point to emphasize here is that the nonlinearity is smooth (i.e., differentiable everywhere), as opposed to the hard-limiting used in Rosenblatts perceptron. A commonly used form of nonlinearity that satisfies this requirement is a sigmoidal nonlinearity defined by the logistic function:

$$y_j \equiv \frac{1}{1 + exp(-s_j)}$$

where s_j is the net internal activity level of neuron j, and y_j is the output of the neuron. The presence of nonlinearities is important because, otherwise, the input-output relation of the network could be reduced to that of a single-layer perceptron. Moreover, the use of the logistic function is biologically motivated, since it attempts to account for the refractory phase of real neurons (Pineda, [18]).

2. The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the network to learn

complex tasks by extracting progressively more meaningful features from the input patterns (vectors).

3. The network exhibits a high degree of connectivity, determined by the synapses of the network. A change in the connectivity of the network requires a change in the population of synaptic connections or their weights.

Indeed, it is through the combination of these characteristics together with the ability to learn from experience through training that the multilayer perceptron derives its computing power. These same characteristics, however, are also responsible for the deficiencies in our present state of knowledge on the behavior of the network. First, the presence of a distributed form of nonlinearity and the high connectivity of the network make the theoretical analysis of a multilayer perceptron difficult to undertake. Second, the use of hidden neurons makes the learning process harder to visualise. In an implicit sense, the learning process must decide which features of the input pattern should be represented by the hidden neurons. The learning process is therefore made more difficult because the search has to be conducted in a much larger space of possible functions, and a choice has to be made between alternative representations of the input pattern.

Research interest in multilayer feedforward networks dates back to the pioneering work of Rosenblatt (1962) [19] on perceptrons and that of Widrow and his students on Madalines [20]. Madalines were constructed with many inputs, many Adaline elements in the first layer, and with various logic devices such as AND, OR, and majority-vote-taker elements in the second layer. The Madalines of the 1960s had adaptive first layers and fixed threshold functions in the second (output) layer [21]. However, the tool that was missing in those early days of multilayer feedforward networks was what we now call back-propagation learning.

The usage of the term back-propagation appears to have evolved after 1985. However, the basic idea of back-propagation was first described by Werbos in his Ph.D. thesis [22], in the context of general networks with neural networks representing a special case. Subsequently, it was rediscovered by Rumelhart, Hinton, and Williams [23], and popularized through the publication of the seminal book entitled Parallel distributed Processing (Rumelhart and McClelland, 1986 [17]). A similar generalisation of the algorithm was derived independently by Parker in 1985 [24].

The development of the back-propagation algorithm represents a landmark in neural networks in that it provides a computationally efficient method for the training of

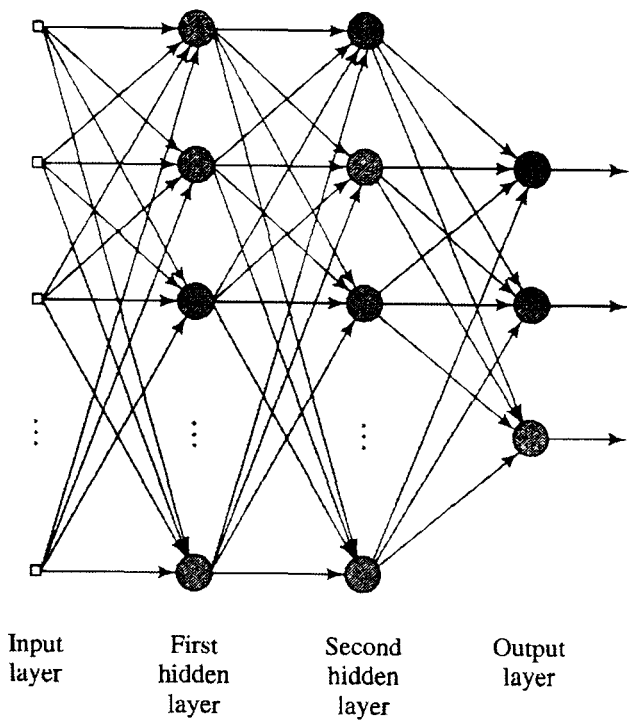


Figure 12. Architectural graph of a multilayer perceptron with two hidden layers.

multilayer perceptrons. Although it cannot be claimed that the back-propagation algorithm can provide a solution for all solvable problems, it is fair to say that it has put to rest the pessimism about learning in multilayer machines that may have been inferred from the book by Minsky and Papert (see [25]).

Derivation of the backpropagation algorithm Figure 12 shows the architectural graph of a multilayer perceptron with two hidden layers. To set the stage in its general form, the network shown here is fully connected, which means that a neuron in any layer of the network is connected to all the nodes/neurons in the previous layer. Signal flow through the network progresses in a forward direction, from left to right and on a layer-by-layer basis.

Figure 13 depicts a. portion of the multilayer perceptron. In this network, two kinds of signals are identified:

1. *Function Signals.* A function signal is an input signal (stimulus) that comes in at the input end of the network, propagates forward (neuron-by-neuron) through the network, and emerges at the output end of the network as an output signal. We refer to such a signal as a "function signal" for two reasons. First, it is presumed to perform a useful function at the output of the network. Second, at each neuron

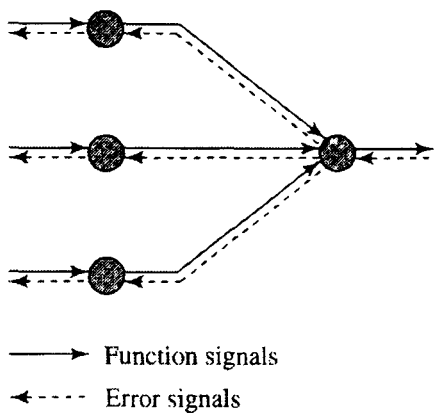


Figure 13. Illustration of the directions of two basic signal flows in a multilayer perceptron, namely, forward propagation of function signals and back-propagation of error signals.

of the network through which a function signal passes, the signal is calculated as a function of the inputs and associated weights applied to that neuron.

2. *Error Signals*. An error signal originates at an output neuron of the network, and propagates backward (layer by layer) through the network. We refer to it as an "error signal" because its computation by every neuron of the network involves an error-dependent function in one form or another.The output neurons (computational nodes) constitute the output layer of the network. The remaining neurons (computational nodes) constitute hidden layers of the network. Thus the hidden units are not part of the output or input of the network hence their designation as "hidden". The first hidden layer is fed from the input layer made up of sensory units (source nodes); the resulting outputs of the first hidden layer are in turn applied to the next hidden layer; and so on for the rest of the network.

Each hidden or output neuron of a multilayer perceptron is designed to perform two computations:

1. The computation of the function signal appearing at the output of a neuron, which is expressed as a continuous nonlinear function of the input signals and synaptic weights associated with that neuron;

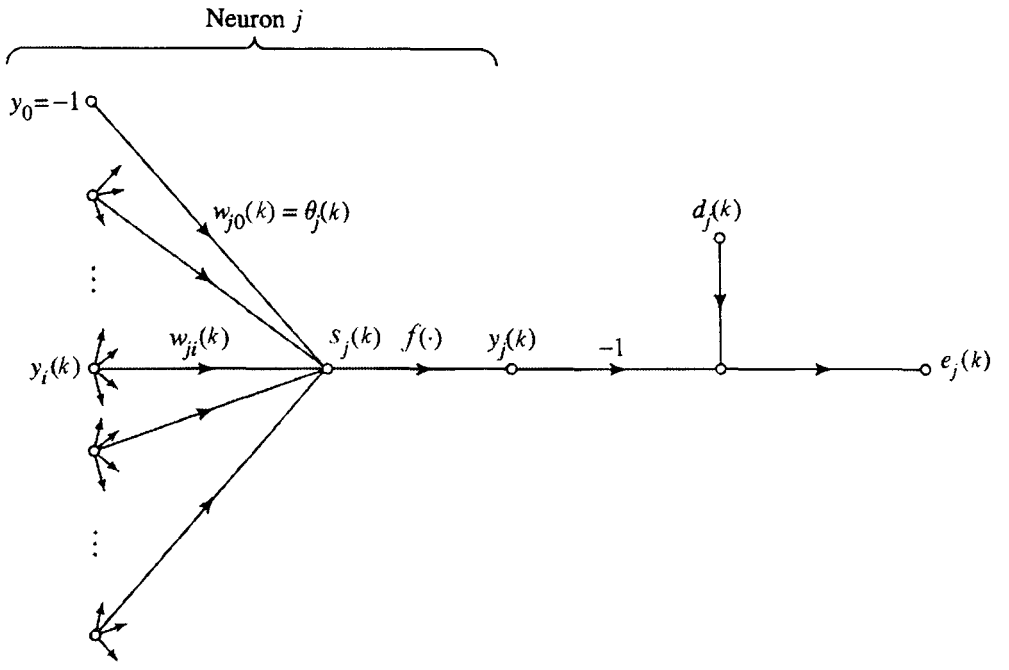


Figure 14. Signal-flow graph highlighting the details of output neuron j

2. The computation of an instantaneous estimate of the gradient vector (i.e., the gradients of the error surface with respect to the weights connected to the inputs of a neuron), which is needed for the backward pass through the network

The error signal at the output of neuron j at iteration k (i.e., presentation of the kth training pattern) is defined by

$$\delta_j(k) = d_j(k) - y_j(k), \text{ neuron } j \text{ is an output node} \tag{12}$$

We define the instantaneous value of the squared error for neuron j as $\frac{1}{2}\delta_j^2(k)$. Correspondingly, the instantaneous value $E(k)$ of the sum of squared errors is obtained by summing $\frac{1}{2}\delta_j^2(k)$ over all neurons in the output layer; these are the only "visible" neurons for which error signals can be calculated. The *instantaneous sum of squared errors* of the network is thus written as

$$E(k) = \frac{1}{2}\sum_{j=C}\delta_j^2(k)$$

where the set C includes all the neurons in the output layer of the network. Let N denote the total number of patterns (examples) contained in the training set. The *average squared error* is obtained by summing $E(k)$ over all k and then normalizing with

respect to the set size N, as shown by

$$E_{av} = \frac{1}{N} \sum_{k=1}^{N} E(k)$$

The instantaneous sum of error squares $E(k)$, and therefore the average squared error E_{av} is a function of all the free parameters (i.e., synaptic weights and thresholds) of the network. For a given training set, E_{av} represents the *cost function* as the measure of training set learning performance. The objective of the learning process is to adjust the free parameters of the network so as to minimize E_{av}. To do this minimisation we use an approximation similar in rationale to that used for the derivation of the *delta-rule* (or LMS) algorithm (subsection 2.1.5). Specifically, we consider a simple method of training in which the weights are updated on a *pattern-by-pattern* basis. The adjustments to the weights are made in accordance with the respective errors computed for each pattern presented to the network. The arithmetic average of these individual weight changes over the training set is therefore an estimate of the true change that would result from modifying the weights based on minimising the cost function E_{av} over the entire training set.

Consider then Fig. 14, which depicts neuron j being fed by a set of function signals produced by a layer of neurons to its left. The net internal activity level $s_j(k)$ produced at the input of the nonlinearity associated with neuron j is therefore

$$s_j(k) = \sum_{i=0}^{p} w_{ji}(k) y_i(k)$$

where p is the total number of inputs (excluding the threshold) applied to neuron j. The synaptic weight w_{j0} (corresponding to the fixed input $y_0 = 1$) equals the threshold θ, applied to neuron j. Hence the function signal $y_j(k)$ appearing at the output of neuron j at iteration k is

$$y_j(k) = f(s_j(k))$$

In a manner similar to the LMS algorithm, the back-propagation algorithm applies a correction $\triangle w_{ji}(k)$ to the synaptic weight $w_{ji}(k)$, which is proportional to the instantaneous gradient $\frac{\partial E(k)}{\partial w_{ji}(k)}$. According to the chain rule, we may express this gradient as follows:

$$\triangle w_{ji}(k) \sim \frac{\partial E(k)}{\partial w_{ji}(k)} = \frac{\partial E(k)}{\partial \delta_j(k)} \frac{\partial \delta_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial s_j(k)} \frac{\partial s_j(k)}{\partial w_{ji}(k)} \tag{13}$$

The correction $\triangle w_{ji}(k)$ applied to $w_{ji}(k)$ is defined by the *delta rule*

$$\triangle w_{ji}(k) = -\eta \frac{\partial E(k)}{\partial w_{ji}(k)} \tag{14}$$

where η is a constant that determines the rate of learning; it is called the *learning-rate parameter* of the back-propagation algorithm. The use of the minus sign in Eq. 2.14 accounts for *gradient descent* in weight space.

Let us introduce the parameter a_j as

$$a_j(k) \equiv -\frac{\partial E(k)}{\partial s_j(k)}. \tag{15}$$

This parameter is called the *activity* of a jth neuron or *local gradient.* The local gradient points to required changes in synaptic weights.

$$\triangle w_{ji}(k) = \eta a_j(k) y_i(k) \tag{16}$$

We may identify two distinct cases, depending on where in the network neuron j is located. In case I, neuron j is an output node. This case is simple to handle, because each output node of the network is supplied with a desired response of its own, making it a straightforward matter to calculate the associated error signal. In case II, neuron j is a hidden node. Even though hidden neurons are not directly accessible, they share responsibility for any error made at the output of the network. The question, however, is to know how to penalise or reward hidden neurons for their share of the responsibility. This problem is indeed the credit-assignment problem. It is solved in an elegant fashion by back-propagating the error signals through the network.

In the sequel, cases I and II are considered in turn.

Case I: Neuron j Is an Output Node

When neuron j is located in the output layer of the network, it would be supplied with a desired response of its own. Hence we may use Eq. 2.12 to compute the error signal $\delta_j(k)$ associated with this neuron; see Fig. 14. Having determined $\delta_j(k)$, it is a straightforward matter to compute the local gradient

$$a_j(k) = \delta_j(k) f'(s_j(k)). \tag{17}$$

Case II: Neuron j Is a Hidden Node

When neuron j is located in a hidden layer of the network, there is no specified desired response for that neuron. Accordingly, the error signal for a hidden neuron would have to be determined recursively in terms of the error signals of all the neurons

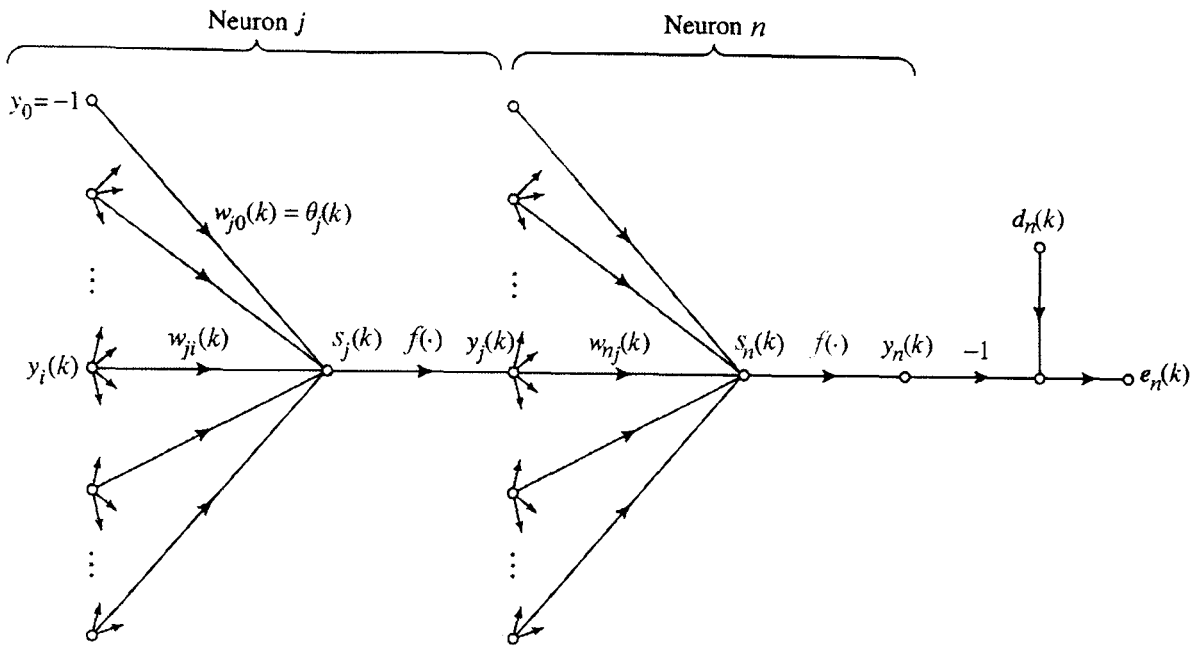


Figure 15. Signal-flow graph highlighting the details of output neuron n connected to hidden neuron j.

to which that hidden neuron is directly connected; this is where the development of the back-propagation algorithm gets complicated. Let's consider the situation depicted in Fig. 15, which depicts neuron j as a hidden node of the network whereas the nth neuron is in the nth output layer. According to Eq. (2.15), we may define the activity for hidden neuron j as

$$\begin{aligned} a_j(k) &= -\frac{\partial E(k)}{\partial y_j(k)} f'(s_j) = -\frac{\partial \left(\frac{1}{2}\sum_{n=C}\delta_n^2(k)\right)}{\partial y_j(k)} f'(s_j) = \\ &= -f'(s_j)\sum_{n=C}\delta_n(k)\frac{\partial(d_n(k)-y_n(k))}{\partial y_j(k)} = f'(s_j)\sum_{n=C}\delta_n(k)\frac{\partial y_n(k)}{\partial y_j(k)} \end{aligned}$$

Taking into consideration that

$$y_n(k) = f(s_n) = f\left(\sum_{j=0}^{p} w_{jn}(k)y_j(k)\right)$$

finally:

$$a_j(k) = f'(s_j(k))\sum_{n=C}\delta_n(k)f'(s_n(k))w_{jn}(k)$$

after rearranging terms, taking into account the Eq. (2.17):

$$a_j(k) = f'(s_j(k))\sum_{n=C} a_n(k)w_{jn}(k) \tag{18}$$

The similar calculations could be done for the subsequent layers, which will give the same result as Eq. (2.18). So, the signal of *activity* of neurons is propagating in back direction and bringing the learning information from the output layer to every previous layer. The signal of activity of neuron a_j depends on whether neuron j is an output node or a hidden node:

1. If neuron j is an output node, a_j equals the product of the derivative $f'(s_j(k))$ and the error signal $\delta_j(k)$, both of which are associated with neuron j; see Eq. (2.17).

2. If neuron j is a hidden node, a_j equals the product of the associated derivative $f'(s_j(k))$ and the weighted sum of the parameters of activity computed for the neurons in the next hidden or output layer that are connected to neuron j; see Eq. (2.18).

The Two Passes of Computation

In the application of the back-propagation algorithm, two distinct passes of computation may be distinguished. The first pass is referred to as the forward pass, and the second one as the backward pass. In the forward pass the synaptic weights remain unaltered throughout the network, and the function signals of the network are computed on a neuron-by-neuron basis.

The backward pass, on the other hand, starts at the output layer by passing the error signals leftward through the network, layer by layer, and recursively computing the a_j (i.e., the *parameter of activity* or *local gradient*) for each neuron. This recursive process permits the synaptic weights of the network to undergo changes in accordance with the delta rule of Eq. (2.16).

Sigmoidal Nonlinearity

The computation of the a for each neuron of the multilayer perceptron requires knowledge of the derivative of the activation function $f(\cdot)$ associated with that neuron. For this derivative to exist, we require the function $f(\cdot)$ to be continuous. In basic terms, *differentiability* is the only requirement that an activation function would have to satisfy. An example of a continuously differentiable nonlinear activation function commonly used in multilayer perceptrons is the sigmoidal nonlinearity, a particular form of which is defined for neuron j by the logistic function

$$y_j(k) = f_j(s_j) = \frac{1}{1 + exp(-s_j(k))}, \qquad -\infty < s_j(k) < \infty \tag{19}$$

According to this nonlinearity, the amplitude of the output lies inside the range $0 < y_j < 1$ Another type of sigmoidal nonlinearity is the hyperbolic tangent, which is

antisymmetric with respect to the origin and for which the amplitude of the output lies inside the range $-1 < y_j < +1$.

Differentiating both sides of Eq. (2.19) with respect to s_j, we get

$$\frac{\partial y_j(k)}{\partial s_j(k)} = f_j'(s_j(k)) = \frac{exp(-s_j(k))}{[1 + exp(-s_j(k))]^2}$$

Since the amount of change in a synaptic weight of the network is proportional to the derivative $f_j'(s_j(k))$, it follows that for a sigmoidal activation function the synaptic weights are changed the most for those neurons in the network for which the function signals are in their midrange.

Thus, we have described above the backpropagation algorithm function. Although many very important aspects of the BP algorithm (such as criteria of optimal speed of learning, optimal number of neurons in hidden layers, etc.) are out of scope of this review, the description will be helpful for understanding of principles of analogue implementation of BP-based ANN described in subsection 3.7.

2.2 Neuromorphic systems: main principles

Since the pioneering work of Carver Mead and his group at Caltech, analogue neural computing has considerably matured to become a technology that can provide superior solutions to many real-world problems.

Many books and technical papers have been written over the last ten years on analogue implementations of artificial neural networks in silicon.

Analogue microelectronic implementations of artificial neural networks offer a number of attractive features:

- they provide an inherent parallelism since computational elements can be highly compact and interconnected;
- in many real-world applications they do not require expensive and bulky analogue to digital converters as they can interface directly to analogue sensory inputs;
- when implemented using weak inversion CMOS circuit design techniques, the resulting system can operate at very low power which could be very attractive for pattern recognition applications in battery operated devices;
- analogue implementations can provide higher operating speeds than digital implementations within their precision and noise margins;
- in fully parallel implementations, they provide a degree of fault tolerance because information is represented in a distributed fashion.
- analogue computational systems operate in asynchronous regime.

However, the attractive features of analogue computation can only be exploited if the following obstacles can be tolerated or overcome:

- accuracy is limited in practice to less than 8 bits which can degrade the performance of learning algorithms;
- noise immunity is a serious design constraint;
- automated analysis and simulation tools of analogue circuits are limited when compared to digital simulation and timing verification tools;

- memory is not accurate and is bulky to implement in up-to-date analogue as well as digital technology.

2.2.1 "Physics for computation"

Biological information-processing systems operate on completely different principles from those with which engineers are familiar. For many problems, particularly those in which the input data are ill-conditioned, biological solutions are *many orders of magnitude more effective* than those we have been able to implement using digital methods. It could be shown that this advantage can be attributed principally to the *use of elementary physical phenomena as computational primitives*, and to the representation of information by the relative values of analogue signals, rather than by the absolute values of digital signals. This approach requires adaptive techniques to correct for differences between nominally identical components, and that this adaptive capability leads naturally to systems that learn about their environment. Although the adaptive analogue systems build up to the present time are rudimentary, they have demonstrated important principles as a prerequisite to undertaking projects of much larger scope (e.g. mulilayer wafer-scale ULSI systems). Perhaps the most intriguing result of these experiments has been the suggestion that adaptive analogue systems are 100 times more efficient in their use of silicon, and they use 10000 times less power than comparable digital systems. It is also clear that these systems are more robust to component degradation and failure than are more conventional systems. The basic two-dimensional limitation of silicon technology is not a serious limitation in exploiting the potential of neuromorphic systems. For these reasons, the large-scale adaptive analogue technology expected to permit the full utilisation of the enormous potential of wafer-scale silicon fabrication.

2.2.2 Analogue multiplier

MOS transistors The MOS (metal, oxide, semiconductor) transistors are main active elements in up-to-date electronic systems. The general form of the MOS transis-

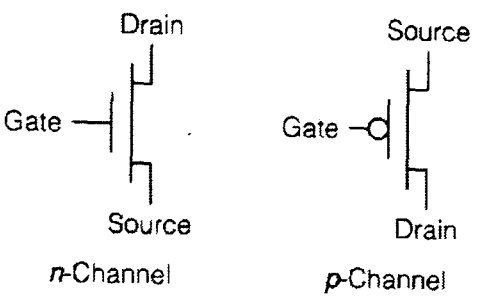


Figure 16. Circuit symbols for $n-$ and $p-$channel MOS transistors.

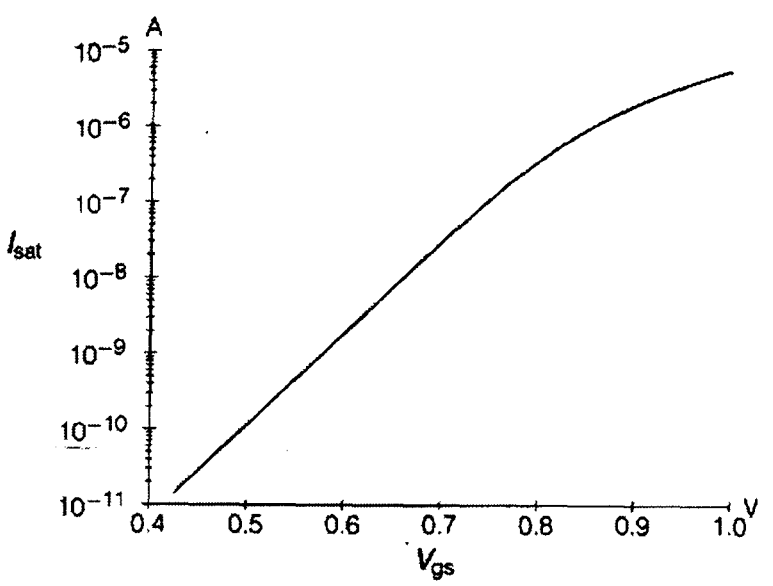


Figure 17. Saturation current of a MOS transistor as a function of gate voltage.

tor current is:

$$I = I_0 e^{-\frac{qV_{gs}}{kT}} \left(1 - e^{-\frac{qV_{ds}}{kT}}\right) = I_{sat} \left(1 - e^{-\frac{V_{ds}}{kT}}\right). \tag{20}$$

$$I_{sat} = I_0 e^{-\frac{qV_{gs}}{kT}}$$

It is possible to see that the dynamic range of the analogue transistor and thus analogue VLSI systems could be rather high: up to $10^6 : 1$ (whereas the dynamic range of natural neurons could be estimated as 100:1.)

Current Operations on current-type signals are well defined; they do not suffer from any of the problems mentioned for voltage-type signals. Current is the flow of charge; the zero of current corresponds to no charge moving. The reference for current

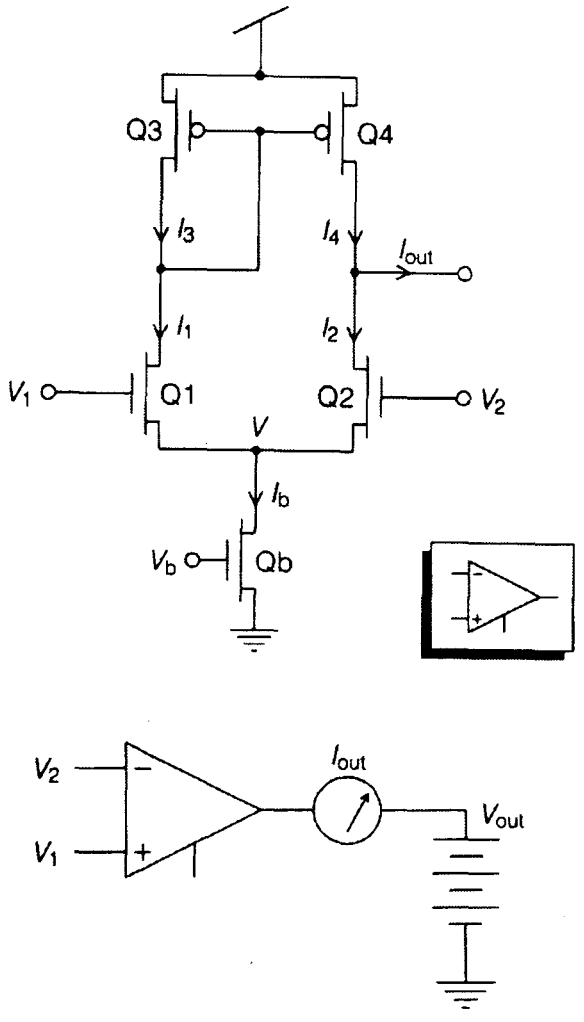


Figure 18. Schematic diagram of the simple transconductance amplifier. The current mirror formed by Q3 and Q4 is used to form the output current, which is equal to I_1-I_2. The symbol used for the circuit is shown in the inset.

is thus the coordinate system within which the transistors or neurons are stationary. We will have no trouble defining a *zero* for current.

Addition and *subtraction* on current-type signals are particularly elegant because they follow from a basic law of physics, Kirchhoffs current law. This Law states that the sum of the currents into an electrical node is zero; that is, the sum of the currents out of the node is the same as the sum of the currents into the node.

Kirchhoffs current law is a result of the basic physical concept of conservation of charge.

Transconductance Amplifier One of the most important elementary circuit is called the transconductance amplifier (an example of the circuit is shown in the Fig. 18.) The transconductance amplifier generates the output current I_{out} that is proportional to

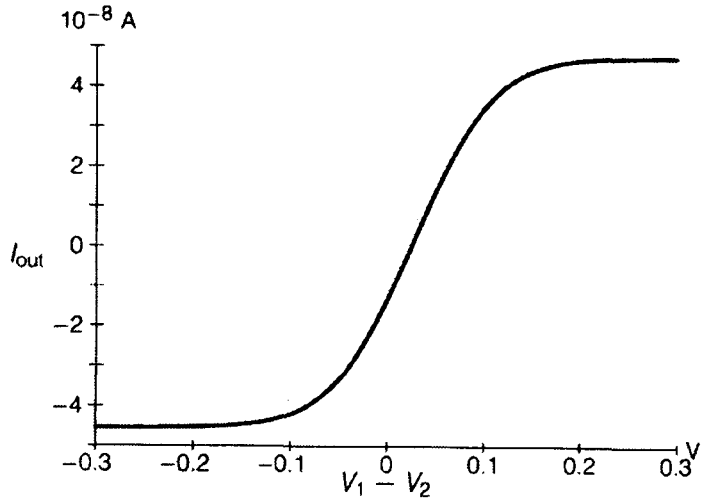


Figure 19. Output current of the transconductance amplifier as a function of differential input voltage. The mismatch between transistor characteristics can be seen in two ways. For this particular amplifier, the input offset voltage is approximately 25 mV, typical for a digital CMOS process. The limiting current for positive inputs is approximately 6% larger than that for negative inputs; a more typical variation would be 20%.

the differential input voltage $V_{12} = V_1 - V_2$ More precise formula for the output current is: $I_{out} = I_1 - I_2 = I_b \tanh \frac{\kappa(V_1 - V_2)}{2}$, $\kappa \sim \frac{0.7e}{kT}$ (e- the electron's charge).

Transistor Mismatch The effects of differences of I_0 between transistors can be seen in Figure 19. The circuit from which these data were taken is typical: The tanh curve is shifted by about 25 millivolts. In addition, the saturated current coming out of Q4 is not the same as the current coming out of Q2. In other words, the negative asymptote is not the same as the positive asymptote. In Figure 19, the difference is about 6 percent.

The Q3-Q4 current mirror does not have 100-percent reflectivity. What we take out of Q3 does not necessarily come out of Q4, because Q4 may have a slightly larger or smaller value of I_0 than does Q3. Differences of a factor of two between I_0 values of nominally identical transistors are observed in circuits such as this. A more typical number for transistors that are physically adjacent is ± 20 percent, corresponding to a difference in gate voltage of ± 10 millivolts.

The difference across a whole chip is not much bigger than a difference between two reasonably closely spaced transistors. The I_0 variation occurs on a small distance scale. For this reason, putting transistors physically close to one another will not eliminate the problem.

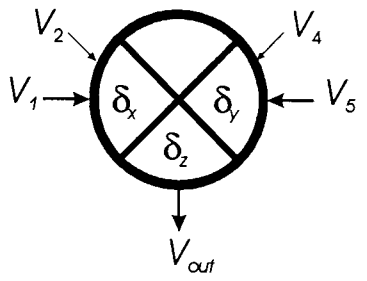


Figure 20. Symbol of analogue multiplier.

We can design circuits in a manner such that these variations can be tolerated. The voltage difference matters in some applications; it does not matter in others. It is a good habit not to trust the transistors to be closer than a factor of two in current (approximately ± 30 millivolts in gate voltage).

The best way to ensure that a circuit will tolerate such variations is to have it *self-compensate for the voltage offsets*.

Self-compensation has another advantage: As circuits age and change and shift with time, the system tunes itself up.

Four-Quadrant Multiplier To multiply in analogue domain a signal of either sign by another signal of either sign, we need a **four-quadrant multiplier**. The principle is illustrated in Figure 21. The output current of the Gilbert multiplier [26] is described by equation:

$$I_{out} = (I_1 - I_2) \tanh \frac{\kappa(V_3 - V_4)}{2} = I_b \tanh \frac{\kappa(V_1 - V_2)}{2} \tanh \frac{\kappa(V_3 - V_4)}{2}.$$

The analogue multiplier is a good demonstration of advantages of analogue technology, since it contain just several transistors whereas the digital multiplier contain hundreds of thousands transistors.

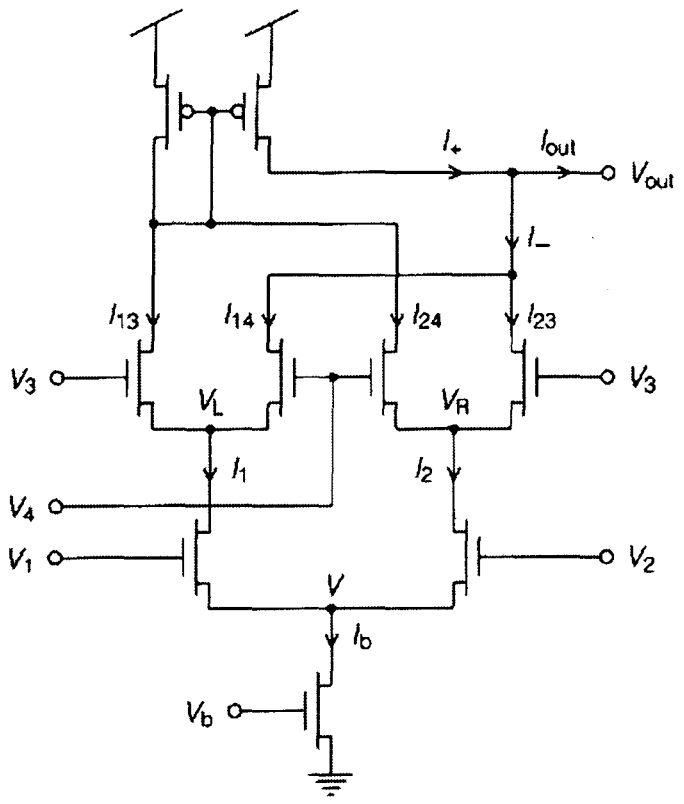
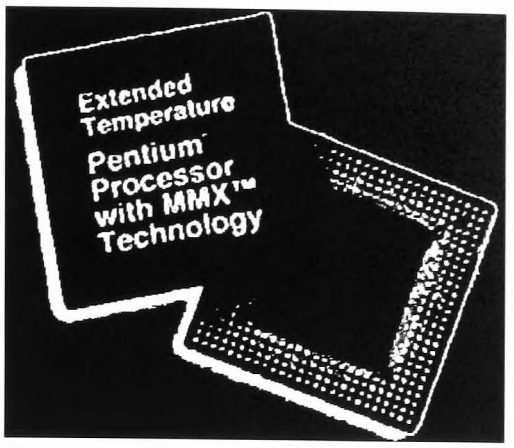


Figure 21. Schematic of the Gilbert multiplier. In the range where $\tanh(x)$ is approximately equal to x, this circuit multiplies $V_1 - V_2$ by $V_3 - V_4$.



Because of the transistors mismatch the output voltage signal of analogue multiplier could be represented by a formula:

$$V_{out} = \delta_z + (V_1 - V_2 + \delta_x) \times (V_3 - V_4 + \delta_y).$$

The symbol of analogue multiplier is shown in Figure 20. The symbols: δ_z, δ_x and δ_y are mismatches of the multiplier.

2.2.3 Other basic elements of neuromorphic systems

We saw (Eq.(2.20)) that a diode-connected transistor creates a voltage that is proportional to the logarithm of the input current. This voltage can be used to control the output currents of other transistors, a technique we used in Figure 22 but it is below the range of usable inputs for circuits such as transconductance amplifiers or multipliers. A voltage that is well within the operating range of these circuits can be generated by two diode-connected transistors in series, as shown in Figure 22(a). The inverse operation creating a current proportional to the exponential of a voltages accomplished by the circuit of Figure 22(b). The relationship between voltage and current for these circuits is shown in Figure 22(c).

From Equation 2.20, we know that the saturated drain current I_{sat} is exponential in the gate-source voltage V_{gs}:

$$I_{sat} = I_0 e^{\kappa V_g} e^{-V_s}$$

Applying this expression to Q1 and Q2, we obtain

$$I = I_0 e^{\kappa V_1} = I_0 e^{\kappa V_2 - V_1}$$

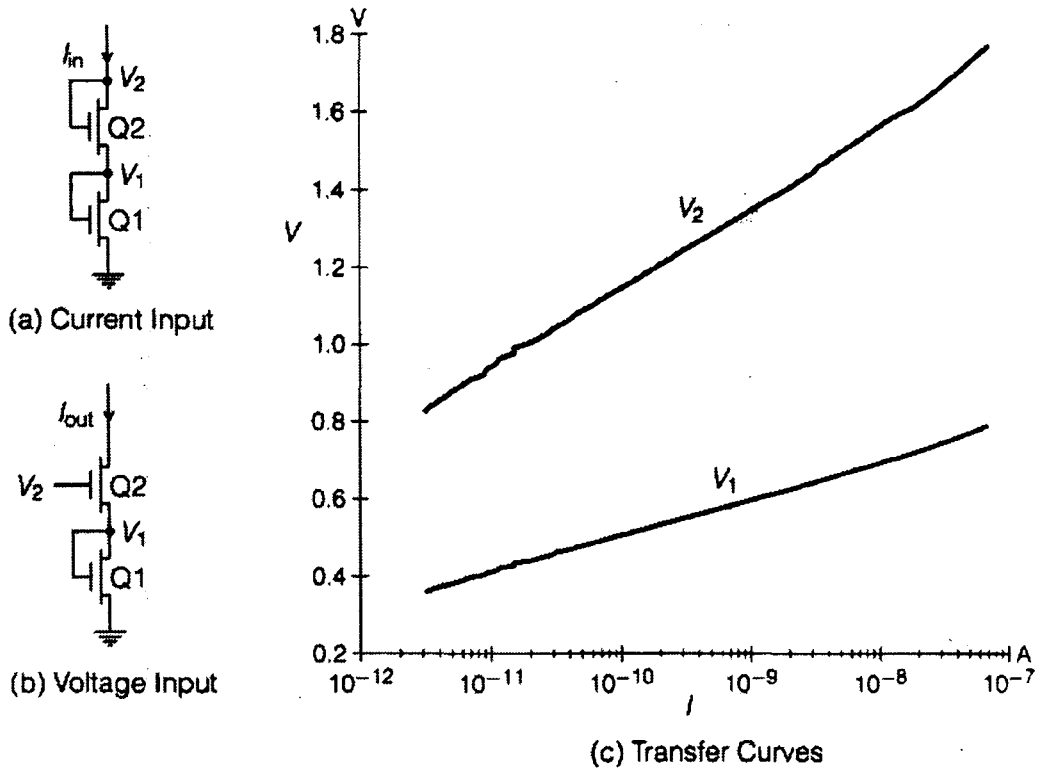


Figure 22. Two circuits that exploit the natural logarithmic voltage-current characteristic of the MOS transistor. The voltage-input version (b) generates an output current that is exponentially related to the input voltage. The current-input version (a) generates an output voltage that is proportional to the logarithm of the input current. The advantage of either arrangement over a single transistor is that the input voltage range is well within the normal operating range of circuits such as the transconductance amplifier. The measured transfer curve for V_2 (c) shows an e-fold increase in current for each 90-millivolt increase in V_2. This value corresponds to $\kappa \simeq 0.7$.

Taking logarithms of the last two terms

$$V_2 = \frac{\kappa + 1}{\kappa} V_1$$

From which we conclude

$$\ln \frac{I}{I_0} = \frac{\kappa^2}{\kappa + 1} V_2$$

In the ideal case the κ is equal to one

$$I = I_0 e^{V_2/2}$$

and we would expect the slope of the upper curve of Figure 22(c) to be twice that of the lower curve.

Summary. We have seen how we can add and subtract currents using Kirchhoffs law, how we can subtract voltages using a differential amplifier, how we can multiply a voltage difference by a current in a transconductance amplifier, and how we can multi-

ply two voltage differences using a Gilbert multiplier. Logarithms and exponentials are primitives provided by the Boltzmann relation. Physics has given us its own natural interpretation of certain mathematical functions. If we can use the primitives nature gives us, we can create formidable computations with physically simple structures. As we evolve the technology to the system level, we will see many applications of this opportunistic principle. The artificial neural system seems to be one of the most promising application.

2.2.4 Pulse-Stream Encoding of information

The pulse-stream encoding mainly implies that the analogue value is encoded in some kind of signal, which composed of pulses of fixed (constant) amplitude and variable duration pulses or period of time between pulses (the exception is the Pulse Amplitude Modulation, which implies that the pulses following with fixed frequency and have modulated amplitude reflecting the variation in V_i - see Figure 23). The advantages of the pulse-stream encoding is in high tolerance of encoded signal to noise, nonlinear distortions, dispersion, decay, etc., which are harmful for analogue signal. On the other hand the converters analogue-to-pulse and pulse-to-analogue are much simpler than ADC and DAC respectively.

Let us look briefly at the pulse stream technique origins. The pulse concept is not new the biological nervous system has been operating on just such a principle for rather a long time for instance [27]. Furthermore, pulse-coding in electrical circuitry is a well-established technique. A condensed review of pulsed techniques in communication is presented in [28], while a more comprehensive treatment is provided by [29], pp. 134-183.

The advantage of the pulse technique is that it allows essentially analogue VLSI devices on a digital CMOS process. Indeed, the digital processes do not incorporate good analogue components such as resistors and capacitors. Furthermore, transistor characteristics are not closely controlled, beyond that which is necessary to maintain correct digital behavior.

Pulse-stream encoding was first used and reported in context of neural integration in 1987 [30],[31].

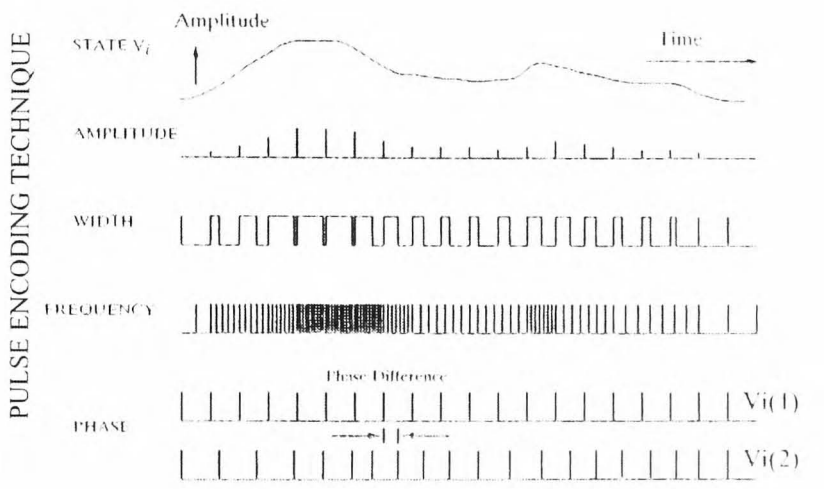


Figure 23. Methods for encoding a time-varying analogue neural state onto a pulsed signal (the picture was taken from [2]).

Pulse Width Modulation This technique is representing the instantaneous value of the state V_i, as the **width** of individual digital pulses (i.e the width is proportional to V_i). The advantages of scheme using the pulse encoding now become apparent, as no analogue voltage is present in the signal and information is coded as described along the time axis. This signal is therefore robust, and furthermore can he decoded to yield an analogue value by integration.

Pulse Frequency Modulation Here, the instantaneous value of the state V_i is represented as the instantaneous **frequency** of digital pulses (i.e. the frequency is proportional to V_i) whose widths are equal. Again, the scheme shows its value, for the same reasons as described above for Pulse Width Modulation.

Pulse or Delay Modulation In this final example, two signals are used to represent each neural state, and the instantaneous value of V_i is represented as the **phase difference** between the two waveforms - in other words by modulating the delay between the occurrence of two pulses on one or two wires. This technique enjoys many of the advantages of the Pulse Width Modulation and Pulse Code Modulation variants described above, but it does imply the use of two wires for signalling, unless one of the signals is a globally-distributed reference pulse waveform. If this choice is made, however, signal skew becomes a problem, distorting the phase information across a large device, and between devices. In a massively parallel analogue VLSI device, signal skew is a fact of life.

Multiplication and summation of pulse signals. The multiplication of pulse encoded signal could be performed by a rather simple scheme, see Figure 24. On a scheme (a) the output current I_{ij} is proportional to the product of the pulse incoded signal V_j and the weight voltage $V_{T_{ij}}$. The output is therefore a stream of current pulses whose amplitude is proportional to $V_{T_{ij}}$ and whose frequency is proportional to V_j.

As Figure 24b shows, the summator for transconductance synapses is composed of an operational amplifier based leaky integrator and a Voltage Controlled Oscillator (VCO). The leaky integrator sums the packets of charge from a column of these transconductance synapses, converting them into the neurons activity voltage. This voltage then controls the duty cycle of the VCO.

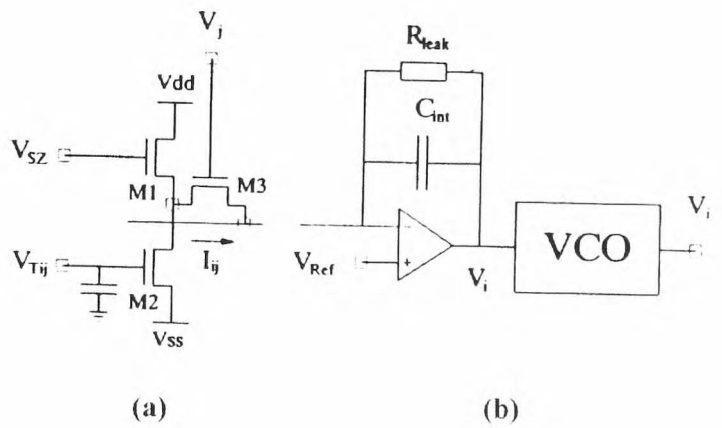


Figure 24. Electrical schemes of a multiplier and summator of a pulse encoded signal.

Since the pulse encoding accumulates some advantages both from digital and analogue encoding, it will definitely be used in the future analogue ANN.

2.3 Examples of neuromorphic systems

First of all let us classify the hardware implementations of artificial neural networks (ANN). Firstly we can divide ANN into two main classes: with off-chip and with on-chip training (see Figure 25).

I. **off-chip learning**:

In this case training is performed completely off the chip and the weights are downloaded to the chip. Such technique implies that the weights are stored into on-chip digital memory. Meanwhile the feedforward network could be either analogue or digital.

Most of the up-to-date neural hardware is based on digital chips implementing the feedforward network of ANN without an on-chip learning system (see Figure 26).

We will mention here the Adaptive Solutions CNAPS [32], Philips L-Neuro chips [33] and Synapse-3 PC board:

- The Adaptive Solutions *CNAPS/PC 128* accelerator card runs at speeds of up to an astounding 2.27 billion connections per second - more than 1,000 times faster than a 486/66;
- *L-Neuro 2.3* is a second generation board that builds on the L-Neuro 1.0 experience. It is cascadable as for the L-Neuro 1.0 and consists of 12 processors that may be operated in either parallel (SIMD) or pipelined modes. It has 16-bit weights and neuron outputs in basic mode. Each of the 12 processors contains 128 16-bit registers for storing weights and states, a 16-to-32 bit multiplier, a 32-bit ALU and a barrel shifter. With a 60MHz clock the chip can compute a 32-bit weighted sum over 12 16-bit inputs every 17ns. This provides 720MOPS or 27 times the 8-bit mode of the L-Neuro 1.0. In learning the 12 weights are updated in parallel within 34ns. ;
- The *SYNAPSE-3* is a low budget PCI-board with 2 MA16 co-processors designed for very fast matrix computations and very fast neurocomputing (Kohonen Self-Organizing Feature Maps and Back Propagation).The peak performance of one board is approximately 2560 MOPS.

We will mention here also two examples of ANN of **analogue** off-chip learning ANN: the Intel Electrically Trainable Artificial Neural Network (*ETANN*) chip [34] and the Edinburgh Pulse-Stream Implementation of a Learning-Oriented Network (*EPSILON*) chip [2].

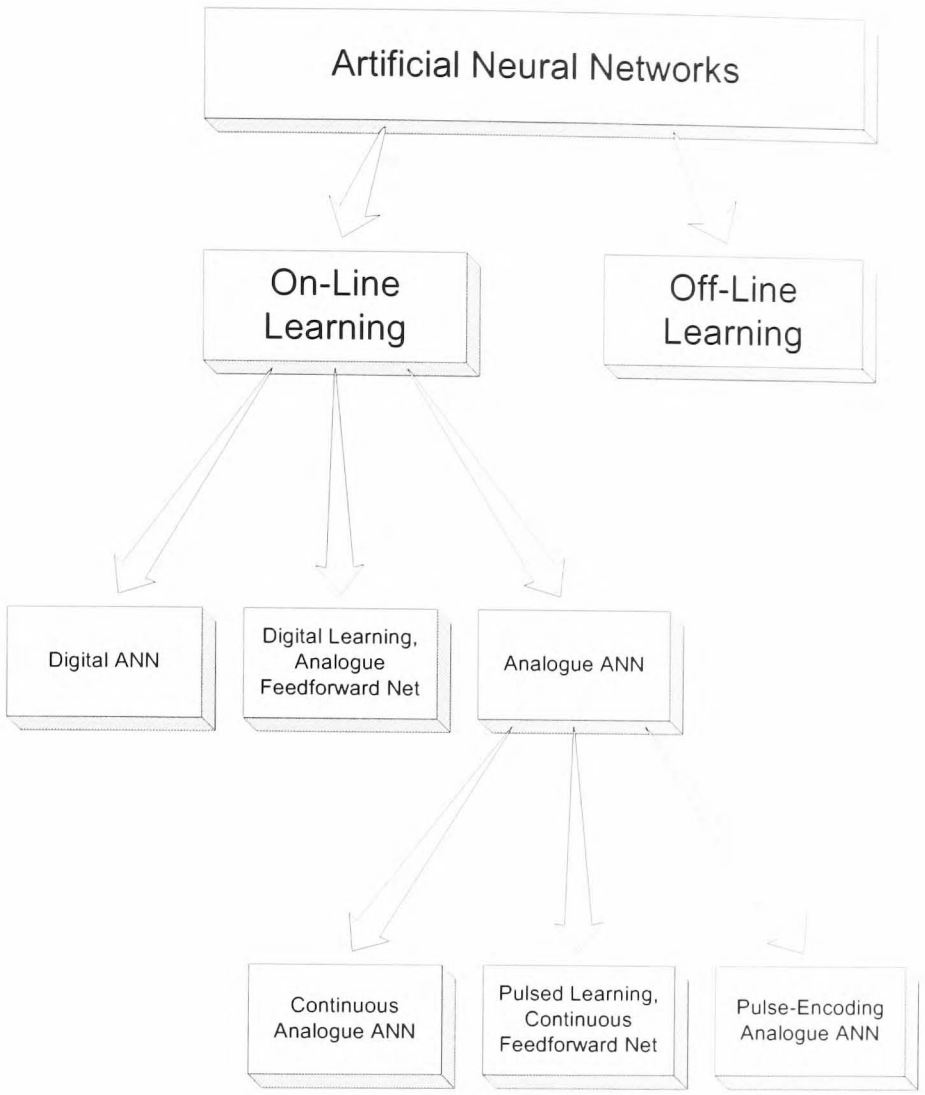


Figure 25. Diagram symbolising the variety of neural hardware implementations.

The ETANN is an example of the continuous analogue computation system, whereas the EPSILON uses the pulse-stream approach to implement the neural architecture.

Such kind of systems (analogue with off-chip learning) could be very useful in some practical applications, where the weights' values could be reliably estimated (during off-chip training) and where the weights' imprecision could be tolerated.

On the other hand in many cases such an approach idoes not work for one of the following reasons:

- the precision of the weight, stored after the off-line training, is not high enough;
- in the case of complex neural networks the learning could be extremely computationally costly, thus off-chip learning on the basis of conventional computing is not applicable (since it could take too long a time) and on-chip learning is necessary;
- the weights, which give a proper result at a given time in the case of analogue systems will not necessarily give the same results in the future (because of temperature sensitiv-

ity and long-term degradation of analogue elements);

- in case of complex ANN their ability to adapt to the variable environment is assumed.

In such cases it is necessary to use systems with

II. **on-chip learning**:

1) The most trivial case is *Digital ANN*. In fact most of such systems are neural simulators rather than neural networks. For example, the use of Digital Signal Processors (DSP) technology gives several orders of magnitude acceleration of the neural network's simulation in comparison with conventional computers. The digital ANN could implement mathematical transforms of any necessary precision. There is no any problem with implementation of highly precision algorithms (such as the "backpropagation" algorithm of learning of Multi-layer Perceptron) in the case of digital ANN

The characteristics of some other neural chips and neural accelerators can be seen in Figures 26 and 27. These tables give us some notion of popular neural hardware implementations. We can see that most of the commercial neural hardware is based on digital technology.

In principle it is possible to imagine also the hybrid neural systems, based on an analogue feedforward network and a digital learning system. In practice, however, such systems are not popular because of incompatibility of the digital elements with high precision analogue ones (resulting into impossibility of securing of a high precision analogue signal if digital elements are in the vicinity of analogue ones). I would say that such (hybrid) systems accumulate the drawbacks of both technologies (analogue and digital) rather than the advantages.

Let us consider now purely analogue ANN. As we can see from Figure 26, purely analogue ANN are not very widespread. The reason is that the backward part of the ANN (the learning system) must be extremely precise (generally speaking the necessary precision of analogue ANN depends on the task the ANN must solve, as well as on a particular neural architecture, but typical necessary precision is about 10^{-6}, see e.g. [51]).

The only known purely analogue MLP ANNs are based on a **perturbation–based learning system**. Such systems are based on either serial weight perturbation (see [52] and [53]) or on the combined outputs and weights perturbation technique (described in [51]). Such perturbation-based systems are very tolerant of analogue nonidealities. On the other hand they are much slower learning, that is a larger number of cycles of learn-

Type	Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
Analog	Intel ETANN	FdFwd, ML	no	6b x 6b	64	10280	2GCPS
	Synaptics Silicon Retina	Neuromorphic	no	na	48x48	Resistive net	na
Digital	NeuraLogix NLX-420	FdFwd, ML	no	1-16b	16	Off-chip	300CPS
	HNC 100-NAP	GP,SIMD,FP	program	32b	100 PE	512K off-chip	250MCPS 64MCUPS
	Hitachi WSI	Wafer, SIMD	Hopfield	9b x 8b	576	32k	138MCPS
	Hitachi WSI	Wafer, SIMD	BP	9b x 8b	144	na	300MCUPS
	Inova N64000	GP,SIMD,Int	program	1-16b	64 PE	128K	870MCPS 220MCUPS
	IBM ZISC036	RBF	ROI	8b	36	64x36	250k pat/s
	MCE MT19003	FdFwd, ML	no	13b	8	off-chip	32MCPS
	Micro Devices MD-1220	FdFwd, ML	no	1b x 16b	1 PE	8	8.9MCPS
	Nestor/Intel NI1000	RBF	RCE,PNN	5b	1 PE	256x1024	40k pat/s
	Philips Lneuro-1	FdFwd, ML	no	1-16b	16 PE	64	26MCPS
	Siemens MA-16	matrix ops	no	16b	16 PE	16x16	400MCPS
Hybrid	AT&T ANNA	FdFwd, ML	no	3b x 6b	16-256	4096	2.1GCPS
	Belkcore CLNN-32	FCR	Boltzmann	6b x 5b	32	992	100MCPS 100MCUPS
	Mesa Research Neuroclassifier	FdFwd, ML	no	6b x 5b	6	426	21GCPS
	Ricoh RN-200	FdFwd, ML	BP	na	16	256	3.0GCPS

Figure 26. Characteristics of some analogue, digital and hybrid neural chips

Type	Name	Chip	Performance
PC Accelerators	AND HNet Transputer 1.0	Transputer T400	na
	BrainMaker Accel.	TI TMS320C25 DSP	40MCPS,500MFLOPS
	Current Tech. MM32k	prop. 2048 PE/chip	4.8MCPS, 2.5MCUPS
	HNC Balbo 860	Intel i860	80MFLOPS
	IBM ZISC ISA	IBM ZISC036	50k pat/sec
	Neural Tech NT6000	TI TMS320C20 DSP	2MCPS
	NeurodynamX XR50	Intel i860	45MCPS
	Nestor Ni1000	Nestor Ni1000	40k pat/sec
	Rapid Imaging 0491E1-ISA	Intel ETANN	2GCPS
	Telebyte 1000 NeuroEng.	prop.	140MCPS
	Vision Harvest NeuroSim.	Intel i860	30MCPS,100MFLOPS
	Ward Sys. NeuralBoard	Inova N64000	5.70GCPS, 1.5GCUPS
	HNC SNAP	HNC 100 NAP	500MCPS, 128MCUPS
	Siemens SYNAPSE-1	Siemens MA-16	800MCPS

Figure 27. Characteristics of some neural PC Accelerators

ing is necessary to reach the same result. For example the serial weight perturbation-based system is N times (N is the number of interconnections, i.e. synapses) slower than the fully parallel backpropagation learning system. In the case of a complex neural network (with e.g. $N = 10^4$) such a drawback of perturbation-based analogue ANNs could make them useless.

If we take a look at natural neural networks, they obviously are based on **parallel-learning**.

Such kinds of systems, which should be highly precise (although they are based on imprecise analogue elements), must employ some kind of self-correction system and procedure. The procedure of self-correction of natural neural networks might be taking place **during the sleep**.

As for artificial systems, until now non of the ANN were based on the on-chip (or on-board) analogue **parallel** learning systems.

The problem of creation of purely analogue parallel-learning ANN is fundamental and important. The last chapter (3.7) of this thesis will be dedicated to this problem.

2.3.1 Silicon Retina

We will describe briefly a specific neuromorphic system: the Silicon Retina, which was proposed by Carver Mead (see e.g. [6]). It is actually a parallel analogue nonlinear adaptive system, capable of real-time image processing. The artificial retina

(see Figure 29), similarly to the natural one (see Figure 28), performs nonlinear 2D space filtering, thus is equalising the brightness and contrast, and increases the efficient dynamic range of an image (i.e. "compressing" the dynamic range).

The electrical scheme of the artificial retina is rather simple. It consists of the resistive network, each node of which contains a pixel element (illustrated in the circular window of Figure 29). This pixel element, which is a silicon model of the triad synapse, consists of a follower-connected transconductance amplifier by which the photoreceptor drives the resistive network, and an amplifier that takes the difference between the photoreceptor output and the voltage stored on the capacitance of the resistive network. These pixels are tiled in a hexagonal array. The resistive network results from a hexagonal tiling of pixels.

The resistive network produces the spatially smooth approximation of the logarithm of image intensity. Thus the output of such system will be the differential signal from the photoreceptor signal (which is the logarithm of image intensity) and the local average signal produced by the resistive network. Roughly speaking, on the output of the system a signal with spatially equalised contrast and brightness is produced.

Although such a system represents a very simplified model of the eye, it has a number of advantages over standard CCD technology-based digital still or video cameras. In particular the "dynamic range compression" in the analogue domain taking place in such systems allows minimising the noise of the signal, which could be critical in cases of very wide dynamic range images, like x-rays or night-vision images.

On the other hand it should be mentioned that such an artificial retina is much more complex than the standard CCD image sensor. The complexity of the system seems to be excessive since normally images are changing relatively slowly (1/20 s), whereas the time of adaptation of the artificial retina could be less than $1\mu s$. The serial (digital or analogue) real-time image (video) processing system seems to be a much more rational solution. An example of such a system is described in the subsection 3.5.

At the same time, the parallel computational structure of the Artificial Retina allows the creation of very fast image enhancement, which could be important for example in processing of super-fast video.

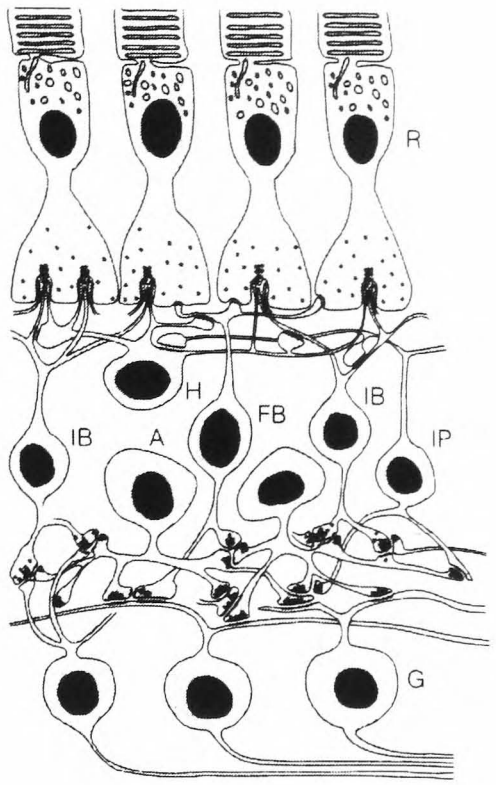


Figure 28. Artist's conception of a cross-section of a primate retina, indicating the primary cell types and signal pathways. The outer-plexiform layer is beneath the foot of the photoreceptors. The invagination into the foot of the photoreceptor is the site of the triad synapse. In the center of the invagination is a bipolar-cell process, flanked by two horizontal cell processes. R: photoreceptor, H: horizontal cell, IB: invaginating bipolar cell, FB: flat bipolar cell, A: amacrine cell, IP: interplexiform cell, G: ganglion cell. (Source: Adapted from [3].)

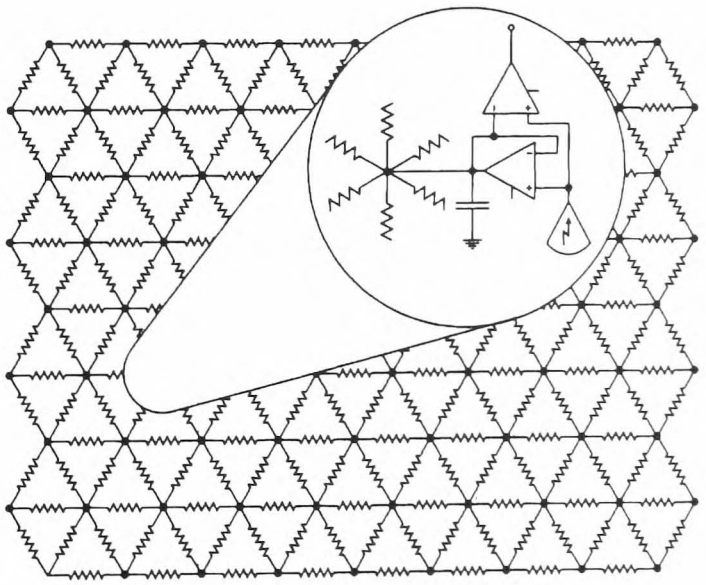


Figure 29. Diagram of the silicon retina showing the resistive network; a single pixel element is illustrated in the circular window. The silicon model of the triad synapse consists of a follower-connected transconductance amplifier by which the photoreceptor drives the resistive network, and an amplifier that takes the difference between the photoreceptor output and the voltage stored on the capacitance of the resistive network. These pixels are tiled in a hexagonal array. The resistive network results from a hexagonal tiling of pixels. (Source: [9])

Chapter 3
Personal results

This chapter is dedicated to analogue adaptive (neuromorphic) systems. The researches were inspired mainly by the fundamental Carver Mead's work [9] and were aimed at the realisation and development of ideas stated in this work.

Most of the systems, described in this chapter, were implemented in hardware as bread-board models. Most of the considered systems are based on polynomial transforms. The principle of orthogonality was also widely employed by the suggested systems.

The final section is dedicated to the problem of creation of purely analogue neural networks with a parallel learning. The unexpected side-effect the work, described in the section, was a very high tolerance of the suggested systems not only to weak imperfections of analogue elements, constituting ANN, but also to damage and degradation of separate elements as well as large areas of ANN hardware.

3.1 Analogue polynomial approximators

In a number of applications the approximation, interpolation or nonlinear extrapolation of certain weakly (when every subsequent term of power series expansion is much less than previous one) nonlinear dependencies $d(x)$, where x an arbitrary signal in time, is necessary. The problem of cancellation of nonlinear distortions of a signal in high precision analogue engineering could be an example of such application. In such cases it seems to be reasonable to use polynomial-based devices.

In this section the neuromorphic devices capable of performing the operations of approximation, interpolation and nonlinear extrapolation are described. The schemes and working characteristics of a breadboard, based on analogue discrete components, are presented. Legendre polynomials were offered as basic functions for significant increasing of the speed of the approximator's convergence.

3.1.1 Power-type-functions-based approximator/interpolator/extrapolator

analogue Polynomial Approximator (APA) is the system, performing the operation of approximation of given two signals' - x and $d(x)$ dependence (see Fig. 30). The signal $x(t)$ can have various values in the range from x_{min} to x_{max}, and the signal $d(x)$ is connected with x by a *simple interrelation*. The aim of the device is to minimise the mean-square deviation of the output signal $f(x)$ from the reference one $d(x)$ under given dependence $x(t)$. The result is achieved by using the procedure of approximation of function $d(x)$ by the polynomial $f(x) = \sum_i W_i x^i$.

The device could function in two regimes: in the regime of training and in the "functional" regime. During the training regime the input signal $x(t)$ and the reference signal $d(x)$ connected with $x(t)$ by a simple interrelation should be applied to related inputs of the APA. In the second regime of the APA function the transfer function of APA is fixed, the signal $x(t)$ is arbitrary and on the output of the APA we have the polynomial approximation of the reference function $d(x)$: $f(x) \simeq d(x)$ (see Figure 31a).

APA is based on neural-like architecture with the "Widrow-Hoff delta rule" algorithm of training [35]. The described device was made on the basis of analogue radio elements and functioning on a principle of continuous (not discrete in time) training. In the regime of training $x(t)$ is a periodical signal with the period T.In the Fig. (31a) the

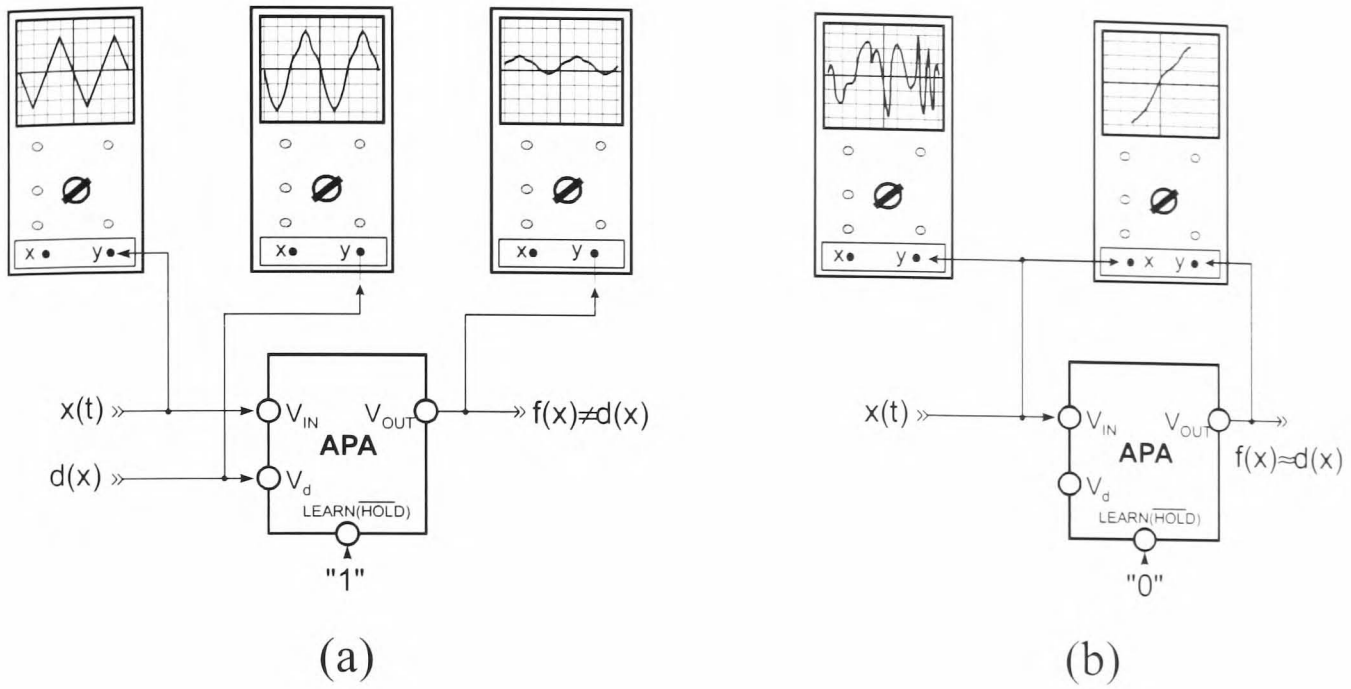


Figure 30. The analog polynomial approximator: (a) - in the regime of training and (b) - in a "functional" regime.

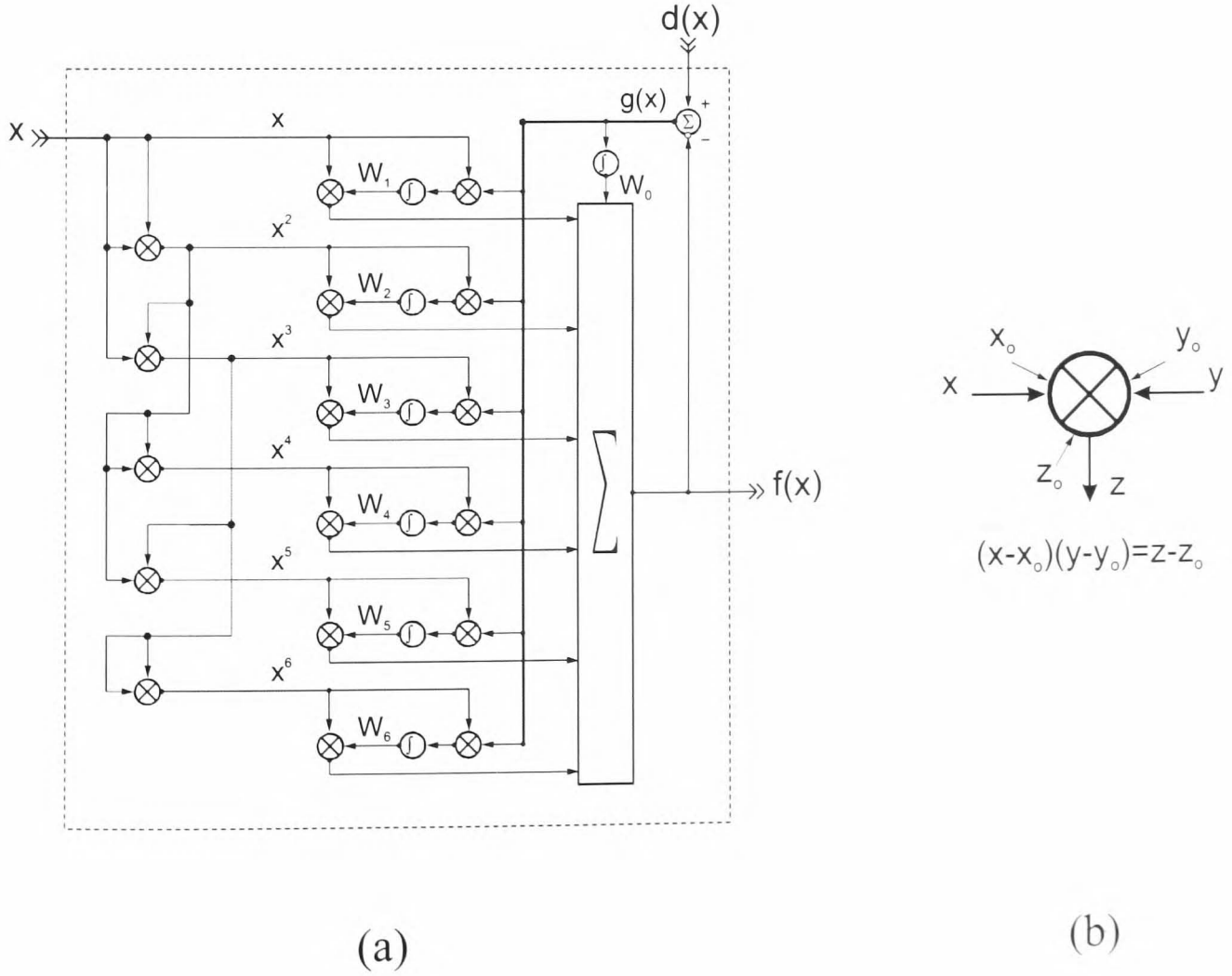


Figure 31. Schematic diagrams: (a):APA in regime of training and (b): Analog multiplier

electrical scheme of the system is presented. The device consists of

1. Synthesizer of power-type functions from the source signal x in accordance with the formula $y_i = x^i$

2. A neuron (an adaptive element) with the linear activation function: $f(x) = \sum_i W_i y_i$.

The "Widrow-Hoff delta rule" training system performing the continuous in time updating of the weights W_i with the purpose of mean-square deviation - e minimisation, where

$$e = \sum_k [d(x_k) - f(x_k)]^2 = \int_T [d(x(t)) - f(x(t))]^2 dt. \tag{21}$$

Assuming that the training is sufficiently slow, that is: $\Delta W_i|_T \ll W_i$ (where T is a period of the training cycle), we can represent the weights updating δW_i in a form:

$$\delta W_i = -\alpha \frac{\partial e}{\partial W_i} \delta t = 2\alpha(d(x) - f(x))x^i \delta t.$$

The $x(t)$ is a periodical signal. If $x(t)$ is a sawtooth waveform signal (see. Fig. 30) than the APA will minimise the *mean square deviation* between **functions** $d(x)$ and $f(x)$. Otherwise the APA will minimise the mean square deviation expressed by a formula (3.21).

The breadboard model basic element is a chip - analogue multiplier (AD734, *Analog Device*), (see Fig. 31b). Its operation can be described by the following equation: $(x + \delta x + x_0) \cdot (y + \delta y + y_0) = z + \delta z + z_0$, where x and y are inputs, z is an output of analogue multiplier, δx, δy and δz are offsets, x_0, y_0 and z_0 are correction inputs. The analysis has showed that for normal function of the system it is not necessary to adjust all offsets: δx, δy and δz in each microchip, but it is enough to adjust only δz in the learning circuit of each cascade of the system. Another offsets will be compensated by previous cascades of the system.

The dependence of signal deviation from the cycle number is shown in Fig. 33a. At the first moment (back edge of a surface) the system is not trained ($f(x) = 0$, for any x). As a result of the system's training the maximal deviation of $d(x)$ from $f(x)$ becomes smaller than 0.1%. It takes approximately 10^5 cycles for the convergency of the system, that is several minutes in the experiment (see Fig. 33c).

The experiments and the analysis have shown that APA can be successfully used also for interpolation and nonlinear extrapolation of various smooth dependencies.

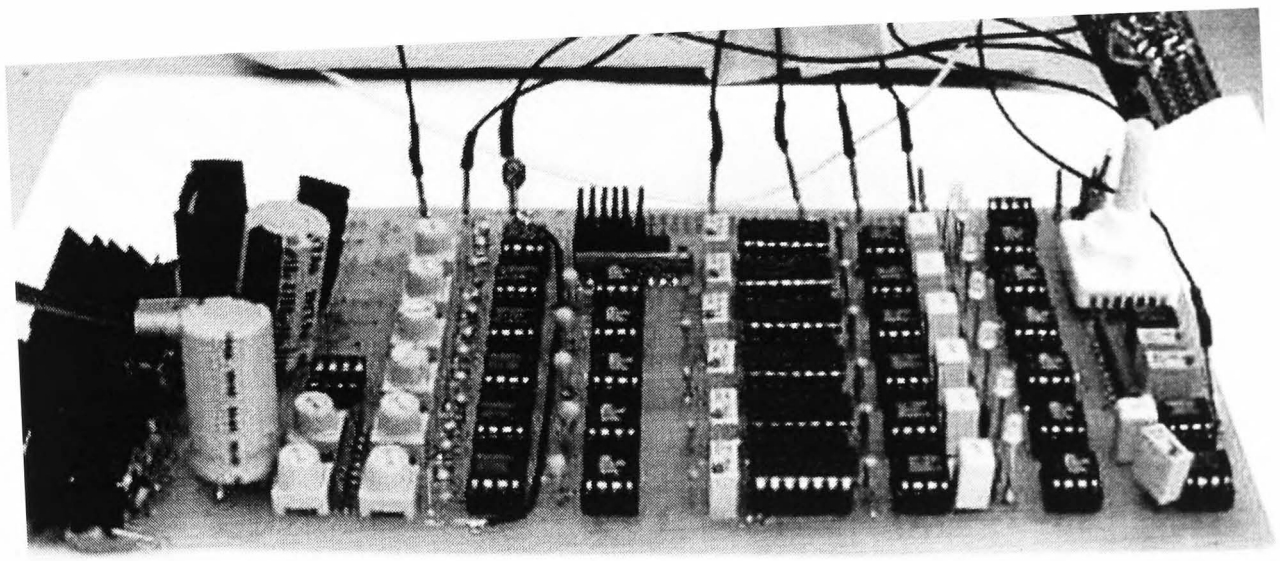
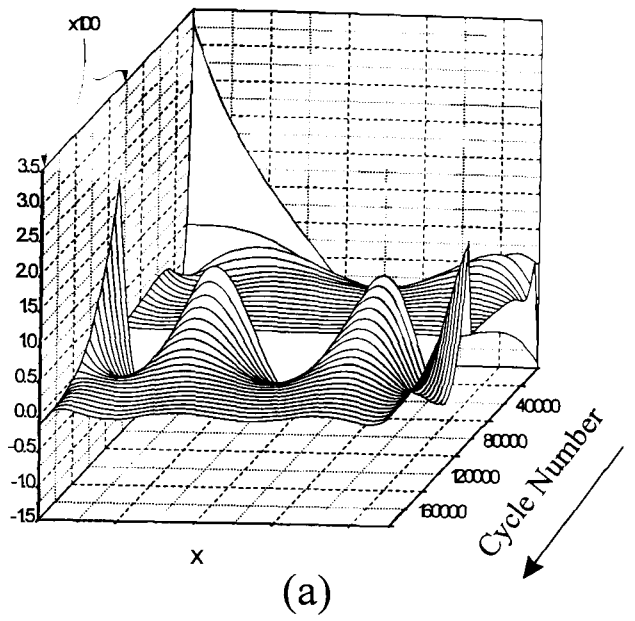
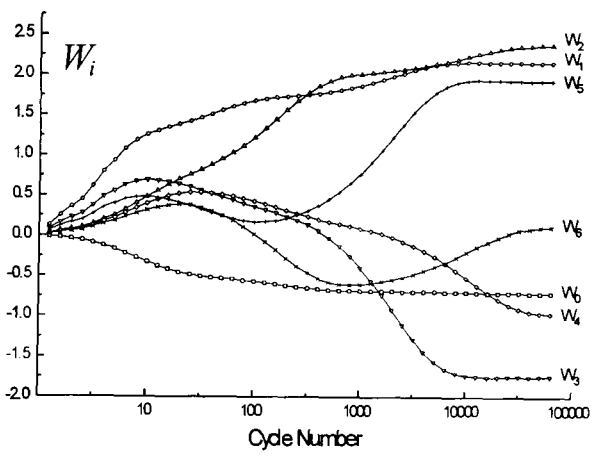


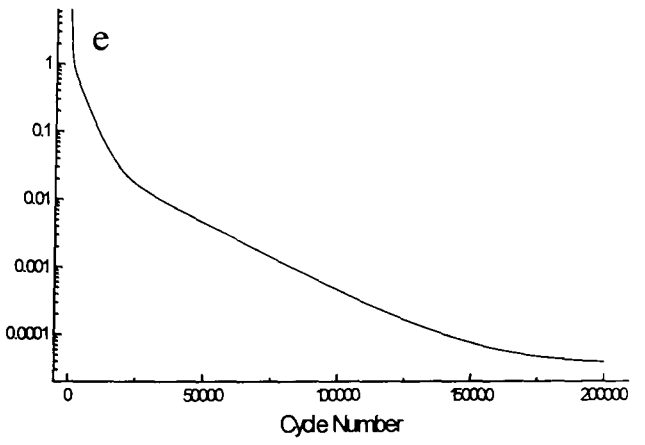
Figure 32. Breadboard of Analogue Polynomial Approximator



(a)



(b)



(c)

Figure 33. Typical dependencies of (a) deviation signal $(d(x)-f(x))$, (b) weights W_i and (c) mean- square deviation signal e as functions of time (number of training cycle) for APA based on power-type functions.

In the Fig. 34a the result of interpolation of a sample dependence $d(x_k)$ (that was generated with addition of noise from the reference dependence $D(x) = 3 + 5x^2 - e^{2x} - e^{-2.2x}$ for values x_k) by the polynomial $f(x)$ is shown.

In the Fig. 34b the result of extrapolation of a sample dependence $d(x)$ by a polynomial $f(x)$ is shown. The sample dependence $d(x)$ is determined on the range $-1 < x < 1$ and equal on this interval to a reference function $D(x)$. The task of the extrapolator is to predict (relying on the $d(x)$ defined on the interval $-1 < x < 1$) the behavior of $D(x)$ on the range $1 < x < 2$ (see the Fig. 34b.)

3.1.2 Fast-training analogue polynomial approximator

As it was shown above, it takes a rather large number of cycles of training (and respectively much time) for convergence of the Analogue Polynomial Approximator. The learning time can be considerably reduced by choosing orthogonal functions instead of the power-type functions as a basis for the function's approximation. In a case of saw-tooth waveform signal $x(t)$ (in regime of training) it will be the orthogonal at the range (x_{min}, x_{max}) Legendre polynomials' basis - $P_i(x)$ (see Fig. 35).

N^o	Power-type functions x^i	Legendre polynomials P_i
0	1	1
1	x	x
2	x^2	$(3x^2 - 1)/2$
3	x^3	$(5x^3 - 3x)/2$
4	x^4	$(35x^4 - 30x^2 + 3)/8$

So the offered system consists of:

a) the synthesiser of Legendre polynomials $P_i(x)$

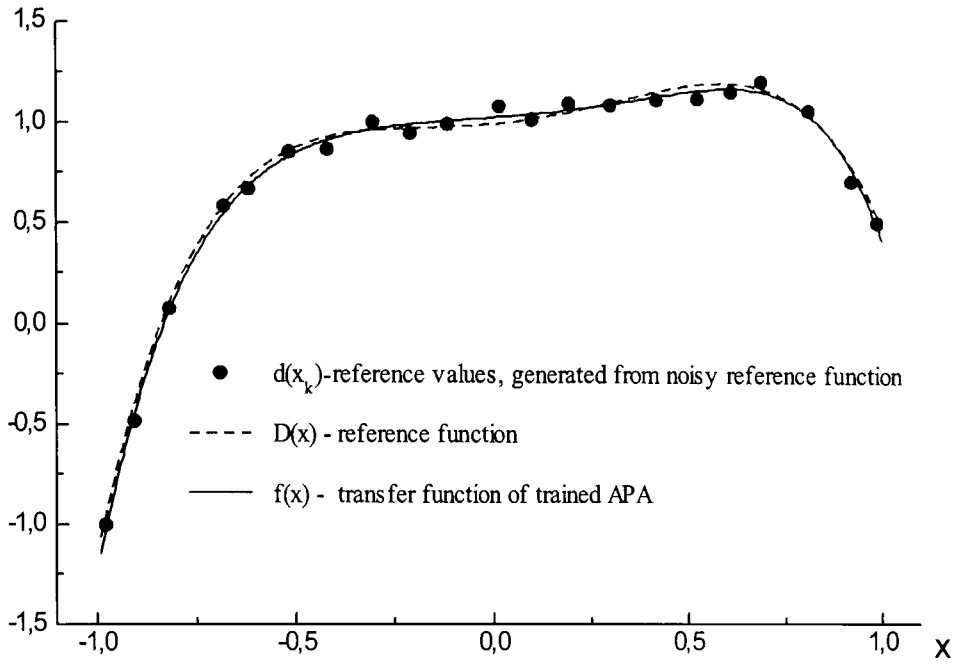
b) the neuron with the "Widrow-Hoff delta rule" algorithm of training:

$$f(x) = \sum_i W_i P_i(x) \tag{22}$$

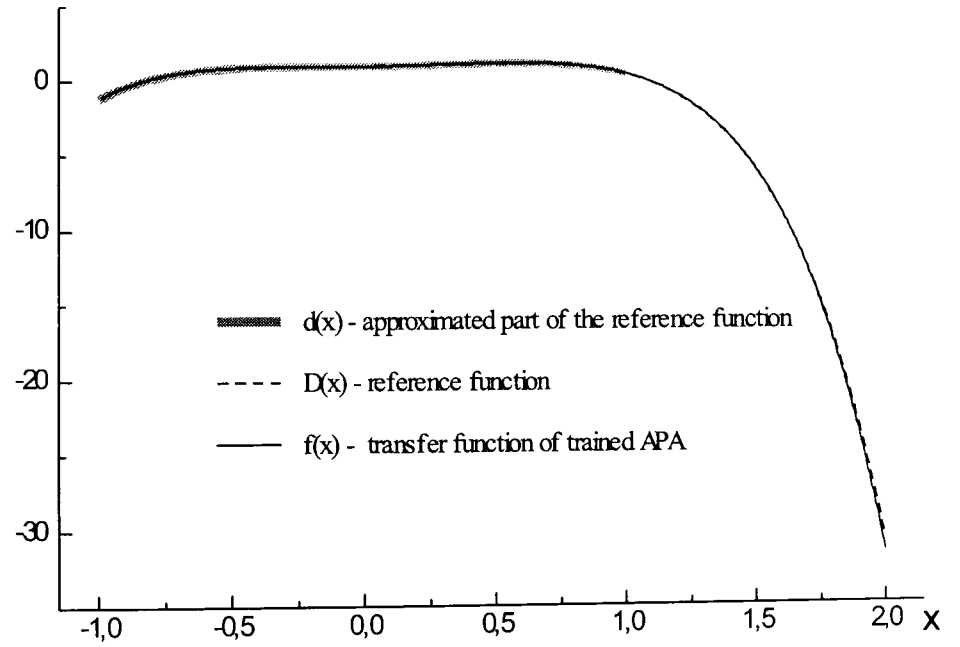
$$\delta W_i = -\alpha \frac{\partial e}{\partial W_i} \delta t = 2\alpha(d(x) - f(x))P_i(x)\delta t. \tag{23}$$

The electrical scheme of the device is shown in the Fig. 36.

It can be easily shown that the use of Legendre polynomial basis (thanks to its orthogonality) allows the *training of each degree of freedom of the system without off-*



(a)



(b)

Figure 34. Examples of APA function in regimes of: (a) interpolation, (b) extrapolation.

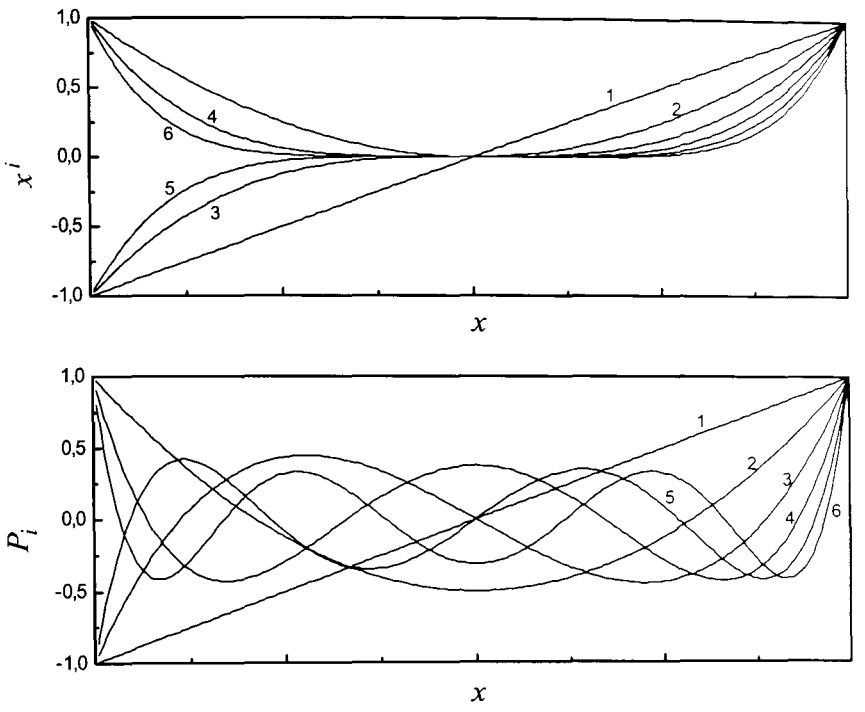


Figure 35. Power-type functions x^i and the Legendre polynomials $P_i(x)$.

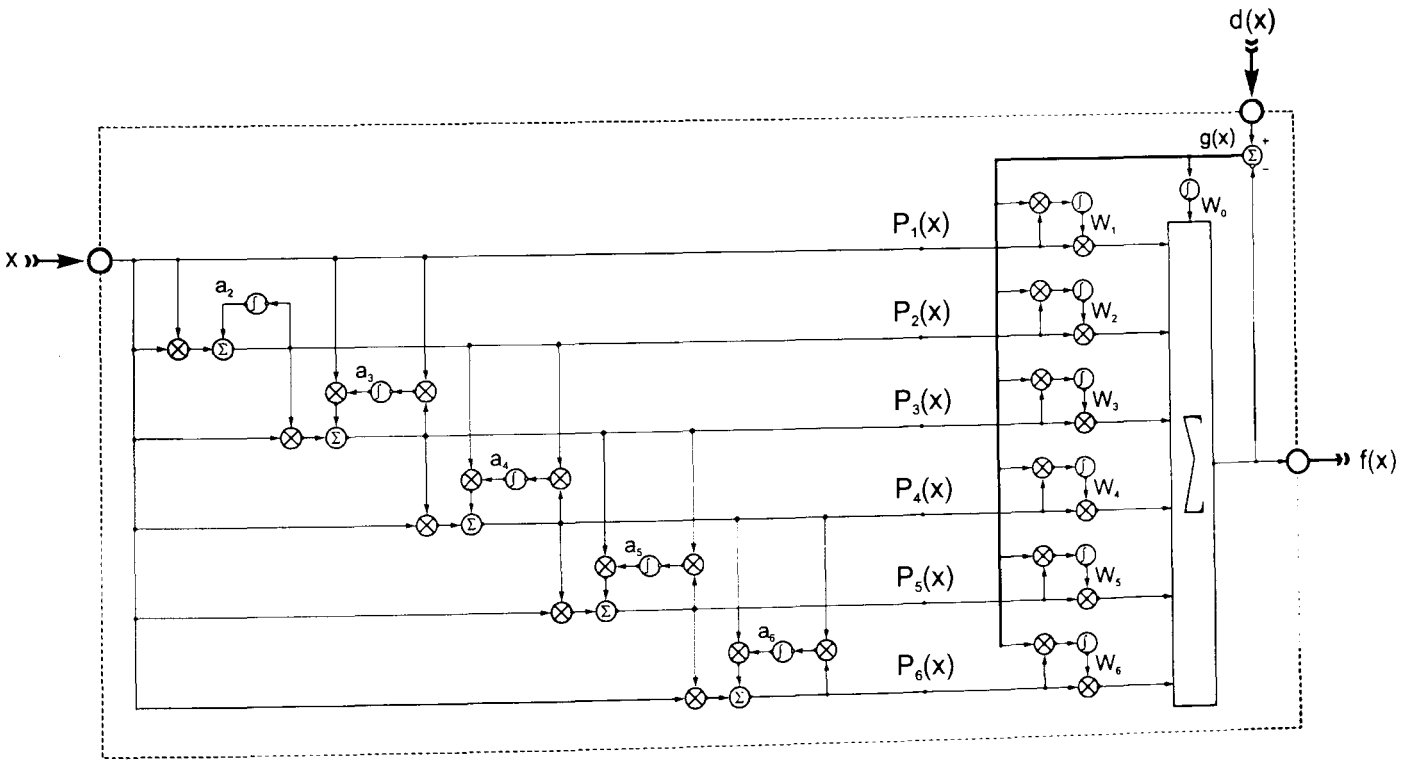


Figure 36. Electrical scheme of the analog approximator on the basis of Legendre polynomials.

tuning of other degrees of freedom. Substituting (3.22) into (3.23)

$$\delta W_i = -\alpha \frac{\partial e}{\partial W_i} \delta t = 2\alpha \left(d(x) - \sum_i W_i P_i(x) \right) P_i(x) \delta t \tag{24}$$

In assumption of slow enough training, i.e. $\Delta W_i|_T \ll W_i$, (T is a period of training circle), taking into account the orthogonality of Legendre polynomials in the case of sawtooth waveform signal $x(t)$, regrouping the Eq.(3.24) and, renaming $2\alpha = \alpha$:

$$\frac{\partial W_i}{\partial t} + W_i \frac{\alpha}{T} \int_T P_i^2(x) dt = \frac{\alpha}{T} \int_T d(x) P_i(x) dt \tag{25}$$

Assuming that $W_i|_{t=0} = 0$, finally, solving the differential equations (3.25) we will get the equations describing the dynamics of convergence of the approximator:

$$W_i(t) \approx \frac{\int_T d(x) P_i(x) dt}{\int_T P_i^2(x) dt} \left(1 - \exp\left(-\frac{\alpha t}{T} \int_T P_i^2(x) dt \right) \right).$$

The Legendre polynomials' synthesiser functions on the basis of a recurrence relation:

$$P_0 = 1, \quad P_1(x) = x, \quad and \quad P_i(x) = 2x P_{i-1}(x) + a_i P_{i-2}(x), \quad for \quad i \geq 2, \tag{26}$$

where a_i are determined from the conditions:

$$\int_{-1}^{1} P_i P_{i-2} dx = 0 \tag{27}$$

It should be noted here that in the Eq.(3.27) the integration variable is x, whereas in conventional analogue integrators the integration variable is time. That's why the sawtooth waveform signal $x(t)$ with the period's duration - T was used to synthesise the Legendre polynomials.

Let us consider how the condition (3.27) could be implemented in analogue hardware. Let us rewrite the Eq.(3.27) for the case of sawtooth waveform signal $x(t)$ and substituting (3.26):

$$\int_{t_0}^{\tau} \left(2x P_{i-1} P_{i-2} + a_i P_{i-2}^2 \right) dt = 0 \tag{28}$$

Since the $P_{i-2}^2(x)$ is a positive function, the condition (3.28) can be maintained by means of a negative feedback in accordance with the following equation:

$$a_i(\tau) = -\beta \int \left(2x P_{i-1} P_{i-2} + a_i P_{i-2}^2 \right) dt, \tag{29}$$

(see the Fig. 36.), where β is a factor of speed of convergency of the synthesiser. After the synthesiser's convergency ($\tau \to \infty$) into the stationary-state we will have:

$$\Delta a_i|_T = \int_T P_i P_{i-2} dx = 0$$

This means that it is possible to satisfy the conditions of recurrence relations (3.26) by a hardware implementing the negative feedback (3.29) and hence P_i is a Legendre polynomial of i degree (shown in the Fig. 35).

The computer's modeling shows that the analogue Legendre polynomials based approximator is much faster trained (about 3 orders of magnitude) in comparison with the approximator based on power-type functions described in the above paragraph (compare Figures 33a,b,c and Figures 37a,b,c).

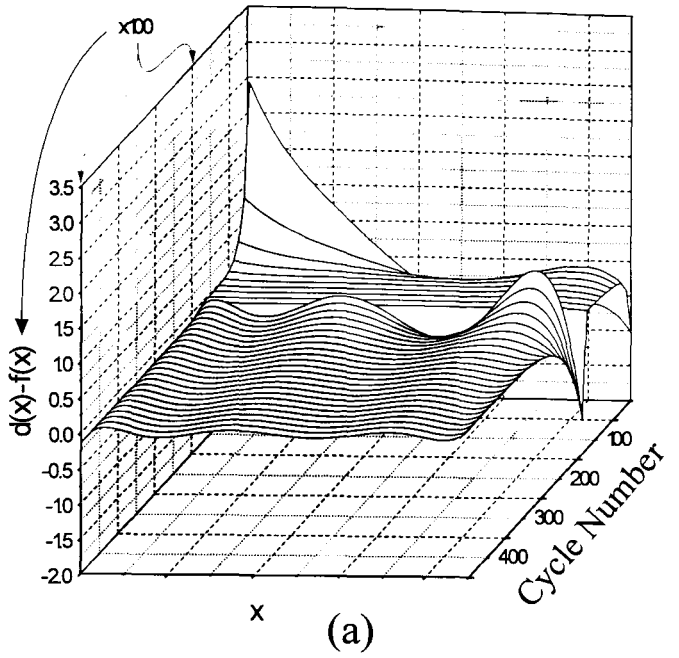
It should be noted, however, that additional circuits of analogue multipliers "zero" correction are necessary for reliable and precise operation of the system represented in Fig. 36 (these circuits are not shown in the scheme).

Conclusions

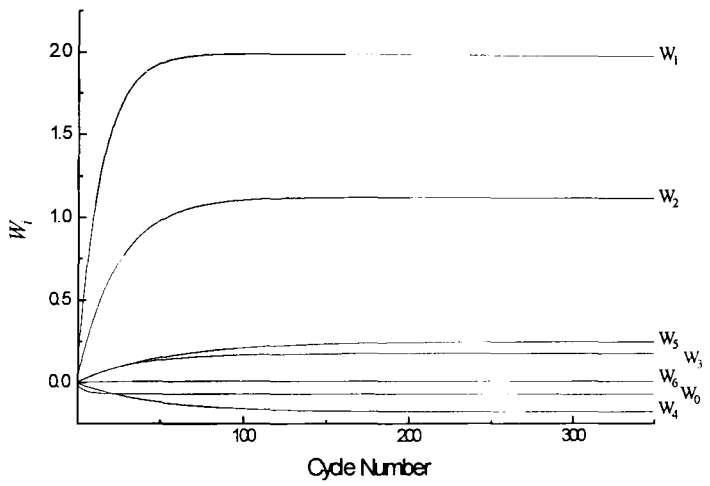
The suggested systems performing functions of approximation, interpolation and extrapolation, were studied experimentally and by methods of computer modeling.

The devices showed good working characteristics (corresponding to theoretical description) In my opinion they can find application in high precision analogue engineering for fast calibration of different devices, as an adaptive nonlinear element capable of compensating nonlinearity of radio-engineering units, etc.

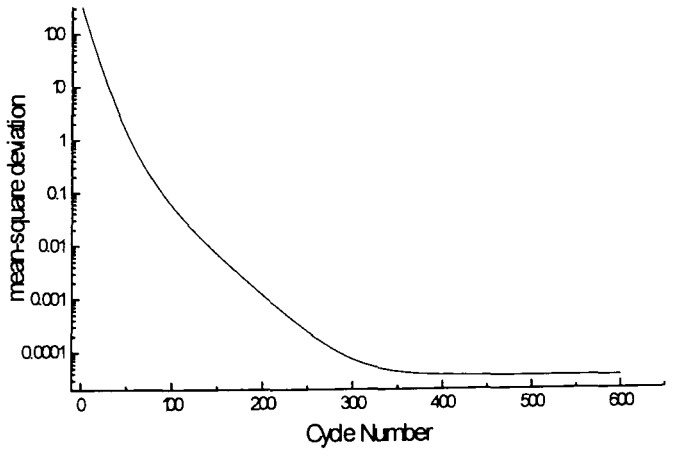
Besides these systems definitely could be pedagogically interesting as simplest adaptive neuromorphic systems. The Legendre polynomials' based approximator is a good demonstration of neuromorphic system using the principle of orthogonality.



(a)



(b)



(c)

Figure 37. Typical dependencies of (a) deviation signal $(d(x) - f(x))$, (b) weights W_i and (c) mean- square deviation signal e as functions of time (number of training cycle) for APA based on Legendre polynomials.

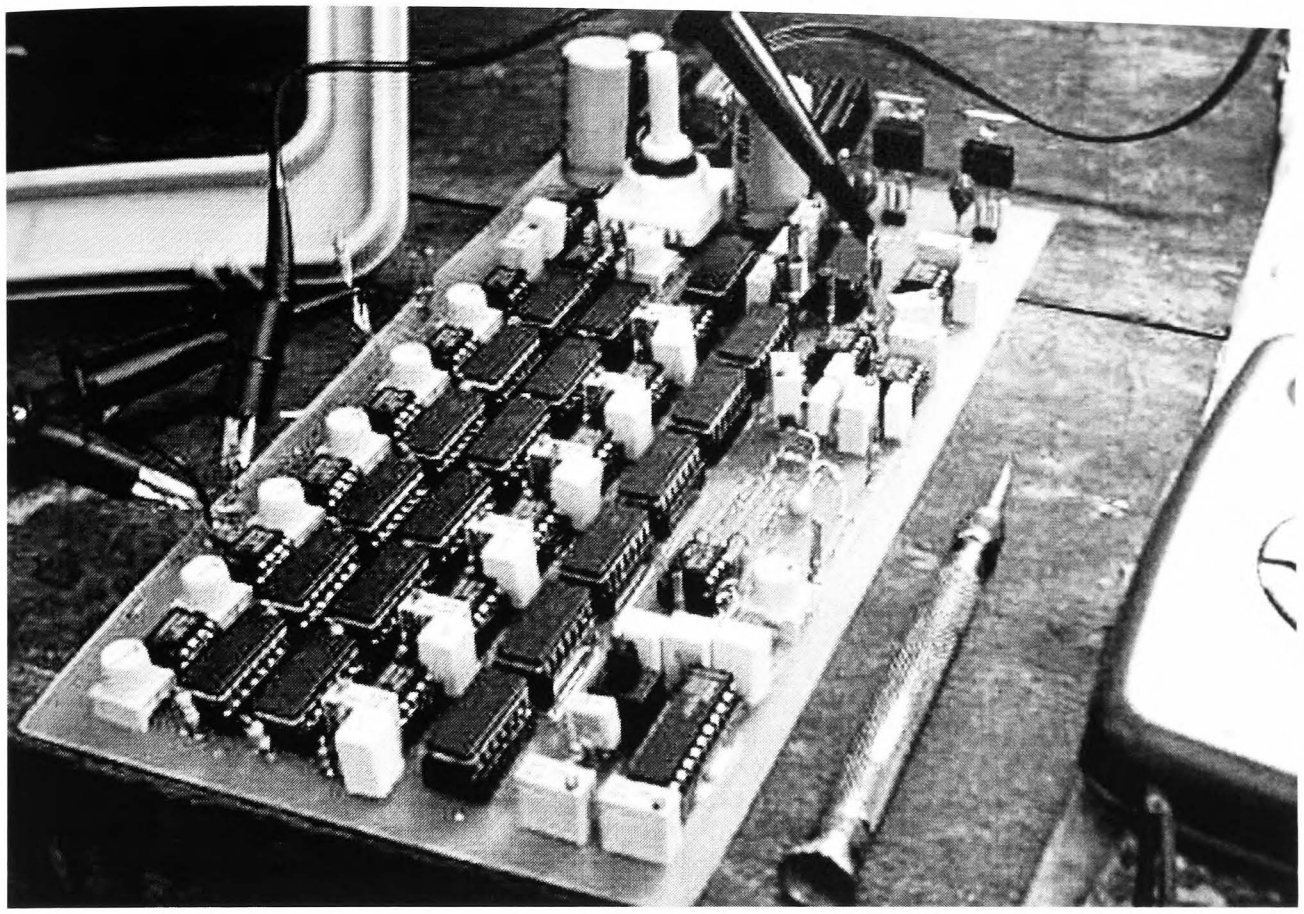


Figure 38. Breadboard of Synthesiser of Orthogonal Signals

3.2 Synthesiser of orthogonal functions

3.2.1 Introduction

Analogue technology for complex and nonlinear information processing permits the creation of devices characterised by several potential advantages as compared with digital devices: greater speed (by more than an order of magnitude), smaller device size and lower power consumption; see e.g. [9] and [36]. The main factor contributing to these potential advantages of analogue technology is the use of nonlinear physical phenomena taking place in transistors to perform the information processing procedures, such as multiplication. For example, typically it takes about 10000 transistors to be switched to perform the operation of multiplication in the digital domain whereas the operation of multiplication can be performed in the analogue domain by a simple scheme of Gilbert's analogue multiplier (see [6]) containing just a few transistors. The analogue implementation therefore takes much fewer transistors. Furthermore, even

more dramatically the statistical length of interconnections is reduced (and it is well known that most power is consumed by wires rather than by semiconductors). This gives us a potential gain in power consumption more than four orders of magnitude.

A more detailed analysis of the potential advantages of the analogue adaptive technology for nonlinear information processing over the digital one can be seen in the very impressive work of Carver Mead [9], which is a basic work of a new and fast-growing direction of neural science and technology: Neuromorphic Systems. It should be noted here that almost a hundred percent of the complex analogue nonlinear systems created up to now are analogue Artificial Neural Networks (references to several developments in the area are available in [37]).

In spite of advantages of analogue technology it is relatively seldom used for complex signal processing. One of the reasons seems to be the fact that the basis of analogue elements (which now includes analogue multipliers, integrators, summators and some elements of nonlinear transform like exponent, logarithm, etc.) is not extensive enough to create purely analogue devices for nonlinear information processing (hybrid analogue-digital integrated systems seem unpromising because they are accumulating disadvantages of both techniques rather than their advantages). Another reason for the restricted use of analogue technology for nonlinear information processing is connected with the disadvantages of analogue elements such as zero off-sets, nonlinear distortions, noise, signal's delay, etc.

Nevertheless there are a lot of potential applications for analogue technology where the parameters of analogue units are quite acceptable. For example it seems to be reasonable to use the analogue technology for creation of devices or elements for voice or image processing and recognition, as well as for some applications in signal synthesis, approximation, interpolation, etc., especially in cases when the data to be processed are noisy. Such devices are supposed to be small-size, fast and low power consuming.

In essence, for creation of up-to-date highly efficient (small size, low power consumption) purely analogue devices for sophisticated signal processing it is necessary to overcome or suppress the disadvantages of analogue technology as well as to increase the number of basic analogue elements.

It should be noted here that the such a disadvantage of analogue technology as zero off-set may be overcome by means of calibration circuits that can be easily imple-

mented in analogue systems. Another very important parameter of elements of analogue system is a scale factor (like as scale factor of analogue multiplier or an amplification coefficient of differential amplifier). The scale factor coefficients of analogue system may be adjusted during the process of analogue VLSI creation. However in this case the system will not be tolerant of temperature changes as well as of a long time degradation of elements. So it can be reasonable in some cases to create special adaptive architectures to solve the problem of scale factor adjustment during the device functioning (or maybe during a special phase of device's self-correction).

In this paper a new analogue system based on adaptive procedure of scale factor coefficients' (in particular, polynomial coefficients) adjustment is proposed. The system is able to produce a plurality of mutually orthogonal signals such as Legendre Polynomials, Cosine Basis of Functions, Smoothed Cosine Basis, etc.

The principle of orthogonality (the use of orthogonal functions and transforms) is a very important paradigm of signal processing. Orthogonal functions are widely used now in digital image processing [38], in systems and control [39], speech processing (see e.g. [40]), in communication (see e.g. [41]), etc. In Author's opinion the creation of analogue VLSI microchip performing the function of orthogonal functions synthesis will permit to increase the spectrum of highly efficient analogue systems of signal processing.

In the paper a proof-of-concept breadboard version of the Synthesiser is described. The process of synthesis of the Smoothed Cosine basis of functions and characteristics of the output functions are described. It is revealed that the device is characterised by a fast and reliable process of signals' synthesis.

3.2.2 Synthesiser

The principal scheme of the Synthesiser of orthogonal signals is shown in Fig. 39b.

The main elements of the device are analogue multipliers, integrators and summators. The device produces orthogonal signals as functions of time: $F_i(t)$, where t is time, $F_i(t)$ is the i-th orthogonal function.

This device (Fig. 39b) will synthesise an orthogonal basis of functions if we will apply to inputs $f(t)$ and $g(t)$ some periodical signals with period T, that are respectively odd and even functions of t in relation to moments of time $t = Tn$ within the ranges of

time $Tn - T/2 < t < Tn + T/2$ (that is, for every n: $f(Tn - \tau) = -f(Tn - \tau)$ and $g(Tn - \tau) = g(Tn + \tau)$, where $0 < \tau < T/2$), n is a cycle number of the periodical (with period T) input (and output) signals.

Furthermore, on the basis of the Synthesiser it is possible to create devices with some useful fixed nonlinear transfer functions, e.g. Legendre polynomials transfer functions, Chebyshev polynomials transfer functions, etc.

Principles of Functioning The output signals of the Synthesiser F_i are the functions of input signals $f(t)$ and $g(t)$ in accordance with the recurrent relations:

$$\begin{aligned} F_0(t) &= g(t); \\ F_1(t) &= f(t)g(t); \\ F_2(t) &= f(t)F_1(t) + a_2F_0(t); \\ &\cdots \\ F_i(t) &= f(t)F_{i-1}(t) + a_iF_{i-2}(t); \end{aligned} \tag{30}$$

where a_i for $i \geq 2$ is defined by:

$$\int_{nI-T/2}^{nI+T/2} F_i(t)F_{i-2}(t)dt = 0; \tag{31}$$

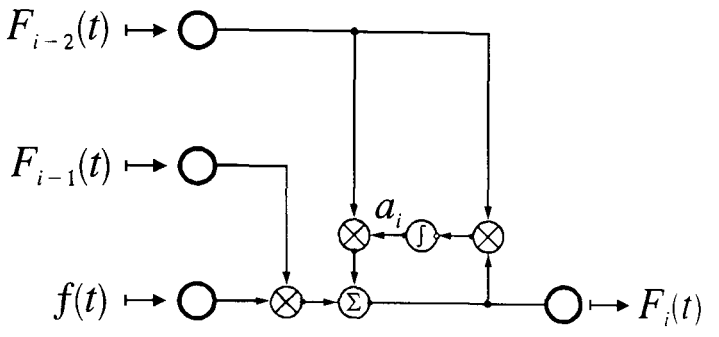
that is

$$\int_{nI-T/2}^{nI+T/2} \left(f(t)F_{i-1}(t) + a_iF_{i-2}(t)\right) F_{i-2}(t)dt = 0; \tag{32}$$

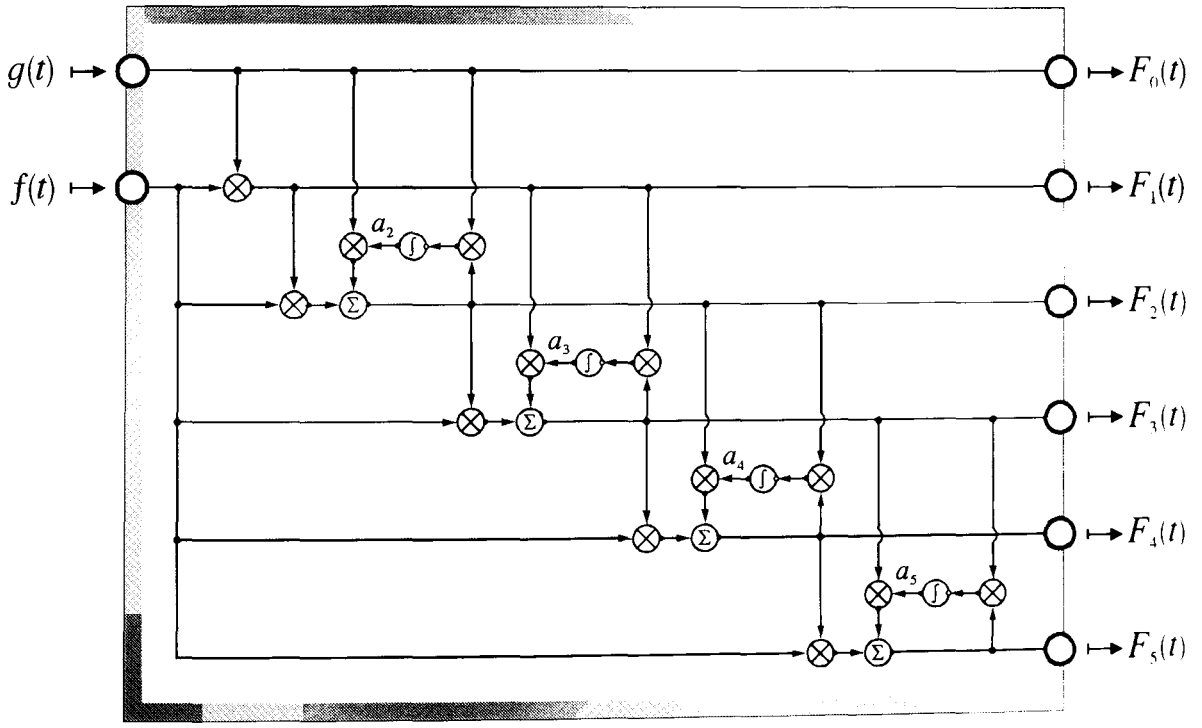
or

$$a_i = -\frac{\int_{nI-T/2}^{nI+T/2} f(t)F_{i-1}(t)F_{i-2}(t)dt}{\int_{nT-T/2}^{nT+T/2} F_{i-2}^2(t)dt}. \tag{33}$$

To realise the condition Eq.(3.31) the device is employing not Eq.(3.33) but the following technique:



(a)



(b)

Figure 39. Scheme of analogue synthesiser of orthogonal signals.

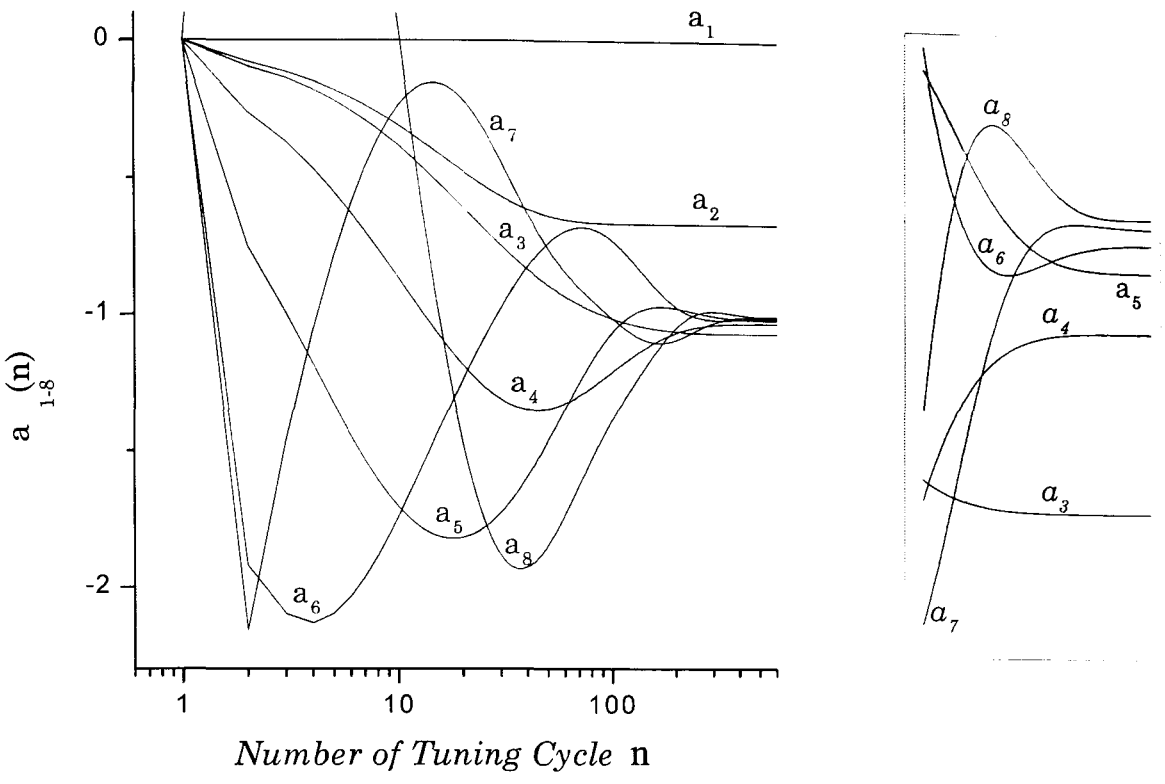


Figure 40. The dynamics of coefficients a_i adjustment (for Legendre Polynomials synthesis). Due to the fact that every coefficient a_i depends only on previous coefficients a_j $(j < i)$ the process of convergence is not only stable, but also very fast.

Due to the fact that $F^2_{i-2}(t)$ is a positive function, the condition Eq.(3.31) can be fulfilled by means of negative feedback in accordance with the following equation:

$$a_i(t) = -\beta \int_0^t F_i(\tau)F_{i-2}(\tau)d\tau = -\beta \int_0^t \left(f(\tau)F_{i-1}(\tau) + a_i(\tau)F_{i-2}(\tau)\right) F_{i-2}(\tau)d\tau \tag{34}$$

(see Fig. 39a and Fig. 39b), where $\beta > 0$ is a Synthesiser's speed of convergence factor. If β is small enough, as a result of the Synthesiser's tuning, in the final stationary state ($\tau \to \infty$, that is: $n \to \infty$) we have: $\Delta a_i|_T \equiv -\beta \int_{nT-T/2}^{nT+T/2} F_i(\tau)F_{i-2}(\tau)d\tau = 0.$

It should be noted here that, due to the fact that every subsequent coefficient a_i depends only on previous ones, the process of adjustment of coefficients a_i is not only stable but also very fast (see Fig. 40).

To imagine the process of adjustment of coefficients $a_i(t)$ let us suggest that at some moment $t = 0$ the first I functions F_i, where $I - 1 \geq i \geq 0$, are already synthesised (hence the a_i coefficients are established and are time-constant) whereas the higher degree functions are not adjusted yet (in particular let's say $a_I(0) = a_I^0$) . Let us analyse in such a situation the behavior of $a_I(t)$ coefficient as a function of time. Let

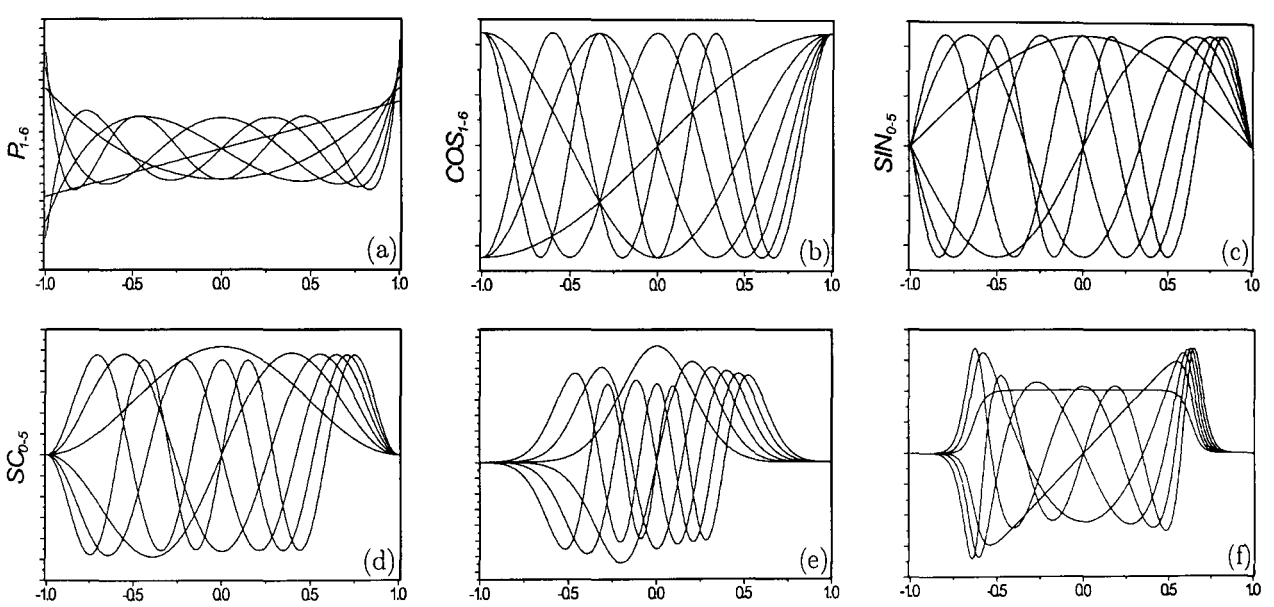


Figure 41. Examples of signals that it is possible to synthesise by the Analogue Synthesiser (see Table 1).

us suggest also, for the sake of simplicity, that β is small enough, so that the $a_I(t)$ coefficient is not changed considerably during one period T of basic functions $f(t)$ and $g(t)$: $\Delta a_I|_T \equiv -\beta \int_{nT-T/2}^{nT+T/2} F_I(\tau)F_{I-2}(\tau)d\tau \ll a_I$. In this case, from Eq.(3.34) for $Tn - T/2 < t < Tn + T/2$ we are getting :

$$\Delta a_I(t)|_T \approx -\beta \int_{nT-T/2}^{nT+T/2} f(\tau)F_{I-1}(\tau)F_{I-2}(\tau)d\tau - a_I(t)\beta \int_{nT-T/2}^{nT+T/2} F_{I-2}^2(\tau)d\tau,$$

and the approximate differential equation for a_I will be:

$$\frac{da_I}{dt} \approx \frac{\Delta a_I(t)|_T}{T} \approx -\frac{\beta}{T} \int_{nT-T/2}^{nT+T/2} f(\tau)F_{I-1}(\tau)F_{I-2}(\tau)d\tau - a_I(t)\frac{\beta}{T} \int_{nT-T/2}^{nT+T/2} F_{I-2}^2(\tau)d\tau.$$

That is:

$$\frac{da_I}{dt} \approx -\beta \left\langle f(t)F_{I-1}(t)F_{I-2}(t)\right\rangle - a_I(t)\beta \left\langle F_{I-2}^2(t)\right\rangle,$$

where $\left\langle f(t)F_{I-1}(t)F_{i-2}(t)\right\rangle$ and $a_I(t)\beta \left\langle F_{I-2}^2(t)\right\rangle$ are expressions for the arithmetic mean of $f(t)F_{I-1}(t)F_{I-2}(t)$ and $F_{I-2}^2(t)$ functions over the period T.

The solution of this equation is

$$a_I(t) \approx \frac{\left\langle f(t)F_{I-1}(t)F_{i-2}(t)\right\rangle}{\left\langle F_{I-2}^2(t)\right\rangle}\left(\exp\left(-\beta \left\langle F_{I-2}^2(t)\right\rangle t\right) - 1\right) + a_I^0 \exp\left(-\beta \left\langle F_{I-2}^2(t)\right\rangle t\right).$$

Taking into consideration that β is a positive constant and that $F_{I-2}^2(t)$ is a positive function, we can conclude that a_I is exponentially converging to stable state

$$a_I \underset{t\to\infty}{=} -\frac{\left\langle f(t)F_{I-1}(t)F_{I-2}(t)\right\rangle}{\left\langle F_{I-2}^2(t)\right\rangle}.$$

We can see that the proposed technique gives finally the same a_i as direct calculation in accordance with Eq.(3.33) but is much more suitable for analogue implementation.

Now, when we have a picture of the synthesis of I^{th} function from $(I-1)^{th}$ and $(I-2)^{th}$ functions, we can approximately imagine the function of the system as a whole. Roughly speaking the dynamic of the synthesiser function can be seen as a subsequent synthesis of functions F_i from the first F_1 to the last function of the synthesiser F_N (N is the number of cascades of the Synthesiser), i.e. F_1, then F_2,... and finally F_N, with the speed of (exponential) convergence proportional to $\beta \left\langle F_{i-2}^2(t)\right\rangle$ for the i^{th} function. This description of the Synthesiser function is in good agreement with the results of

Table 1. Basic functions for synthesis of different orthogonal bases shown in Fig. 41.

	$f(t)$	$g(t)$
a) Legendre polynomials	$\mathrm{Saw}(t)^*$	1
b) Cosine Basis†	$\sin(\pi t/2)$	1
c) Sine Basis	$\sin(\pi t/2)$	$\cos(\pi t/2)$
d) Smoothed Cosine Basis	$\sin(\pi t/2)$	$\cos^2(\pi t/2)$
e) Hermite polynomials	$\mathrm{Saw}(t)$	$\exp\left[-16\,\mathrm{Saw}^2(t)\right]$
f)	$\mathrm{Saw}(t)$	$\frac{1}{1+\exp[-(5\,\mathrm{Saw}(t))^2-10]}$

* $\mathrm{Saw}(t)$ is a Saw-waveform signal.
† In this case the output functions $F_i(t)$ will constitute the basis of Chebyshev polynomials of the first kind $F_i(t) = T_i(t)$.

computer modelling shown in Fig. 40 as well as with experimental results which we have observed with the breadboard of the Synthesiser described in the next section.

The main statement of this section (the proof can be found in an Appendix A) is the following:

if: $f(t)$ and $g(t)$ are some periodical signals with period T , that are respectively odd and even functions of t in relation to $t = Tn$ in every period n (that is, for every n: $f(Tn - \tau) = -f(Tn + \tau)$ and $g(Tn - \tau) = g(Tn + \tau)$, where $0 < \tau < T/2$), and F_i are defined by the recurrent equations (3.30), where a_i are defined by integral equations (3.31),

then: F_i are mutually orthogonal functions.

The most important signals for the orthogonal signals creation are saw-tooth and sine waveform signals as well as some simplest non-linear transforms of these signals based on multiplication, dividing, logarithmic, exponential, etc. transforms which can be performed in analogue domain (like e.g. $\exp(-x^2)$). As an example of an element performing such a nonlinear transform a logarithmic amplifier AD640 (Analog Device) can be considered. The element provides logarithm transform of analogue signal with bandwidth up to 120 MHz with the dynamic range 50 dB and can be used to implement the exponential or the logarithmic transform. Several examples of orthonormal bases of functions are shown in Fig. 41. The basic functions $f(t)$ and $g(t)$ for the synthesis of the orthogonal bases that shown in Fig. 41 are described in Table 1.

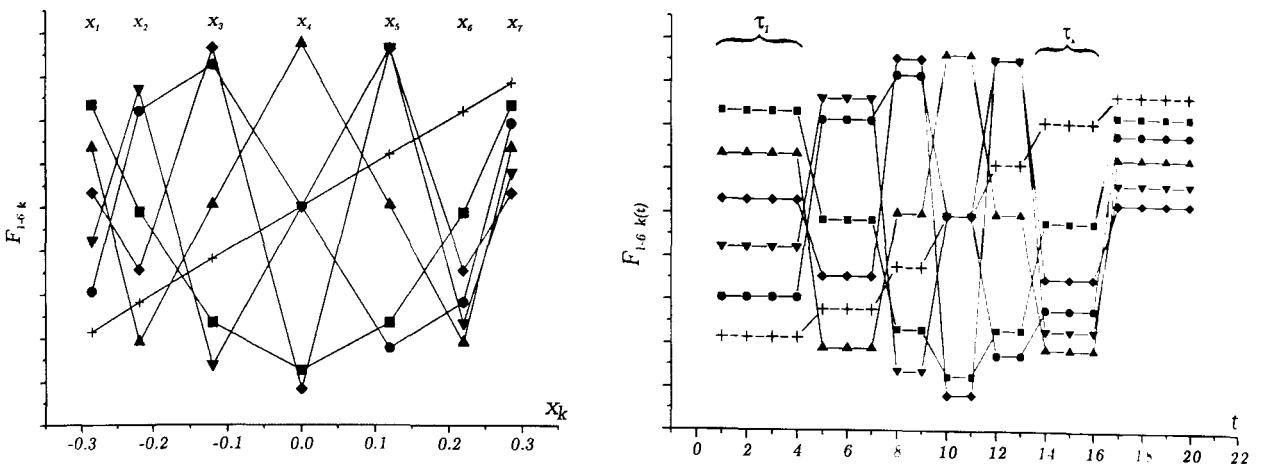


Figure 42. An example of discreet orthogonal sequences $F_{i,k}$ synthesised on the basis of f_k sequence ($F_{1,k} = f_k$, $g_k = 1$).

3.2.3 Synthesis of orthogonal signals discrete in time

The Synthesiser is able to produce not only orthogonal continuous signals but also some discrete orthogonal time sequences.

Let us suggest we have some periodical signals $f(t)$ and $g(t)$, every period of which is consist of M discrete values f_k and g_k with duration τ_k (k is an integer number from 1 to M) so that $f_k = -f_{M-k+1}$, $g_k = g_{M-k+1}$ and $\tau_k = \tau_{M-k+1}$.

If we apply such odd and even discrete signals respectively to inputs f and g of the synthesiser we will have a mutually orthogonal time sequences on the outputs F_i.

An example of such orthogonal time sequences is shown in Fig. 42. In this case $g_k = 1$ and f_k is coincide with $F_{1,k}$ (where $F_{1,k}$ is a signal on the output F_1 of the synthesiser).

Such orthogonal sequences may be useful for example for fast interpolation of some discrete data.

3.2.4 Synthesis of Chebyshev and Legendre polynomials

In the case of input signals $f(t) = \sin(\pi t/2)$ and $g(t) = 1$ on the output we will have the Cosine basis of functions (Fig. 41b): $F_m(t) = \cos\left(\frac{\pi m}{2}(t+1)\right)$.[2] In the stationary state (after the process of adaptation) we will have the transfer functions of

[2] Discrete analogue of Cosine Basis of functions is used for well known Discrete Cosine Transform (DCT) and has already found wide applications in signal and image processing, in particular for JPEG algorithm of image compression.

the Synthesiser as $F_m(\sin(\pi t/2)) = T_m(\sin(\pi t/2))$, where $T_m(x)$ is the m-th degree Chebyshev polynomial of the first kind from the argument $x(t) = \sin(\pi t/2)$. If we then fix the coefficients a_m we will get the device with the fixed transfer function $T_m(x)$, where $x(t)$ is an arbitrary input signal on the input f of the Synthesiser.

Analogously it is possible to create the device with Legendre polynomials' transfer functions $F_m(x) = P_m(x)$ (Fig. 41a). In this case in the adaptive phase it is necessary to apply the sawtooth waveform signal $f(t)$ =SAW(t) to the input f and a constant signal (e.g. $g(t) = 1$) to input g. The description of Legendre, Chebyshev, etc. polynomials as well as some other orthogonal functions is available in [42].

3.2.5 Experiment

A breadboard of the Synthesiser of orthogonal signals (implementing the scheme shown in Fig. 39) has been created. The scheme was realised on the basis of analogue microchips: analogue multipliers AD734AQ (Analog Devices) and operational amplifiers OP27E (Analog Devices). The voltage range of signals of the Synthesiser is from $-10V$ up to $10V$. The power supply voltages are $\pm 15V$. The only difference of the breadboard's scheme from the scheme shown in Fig. 39 was some additional normalising amplifiers in every cascade of the system (not shown in Fig. 39). The breadboard device doesn't contain circuits of self-calibration. Typical offsets of analogue multipliers AD734 are $\delta x, \delta y, \delta z \sim 0.3\%$ (where offsets δx, δy, δz are defined as constants from an equation that approximately describes the transfer function of analogue multiplier: $z = \delta z + (x - \delta x) \times (y - \delta y)$, x and y are inputs of a multiplier and z is an output). The inverting integrators of the Synthesiser are created on the basis of the standard scheme comprising operational amplifier, capacitor in the feedback and resistor.

The aim of the experiment was an investigation of different properties (noise, speed, precision, etc.) of the Synthesiser. In particular it was interesting to investigate experimentally the process of adjustment of coefficients a_i. It was revealed that the process of convergence is very fast: approximately 300 cycles (that is 300 periods of periodical input signals) and stable, and is in good agreement with the results of computer modelling shown in Fig. 40.

The frequencies of basic signals (saw-waveform signal or sine wave) used in experiments were $\leq$ 100 kHz (approximately 1 MHz bandwidth of highest frequency

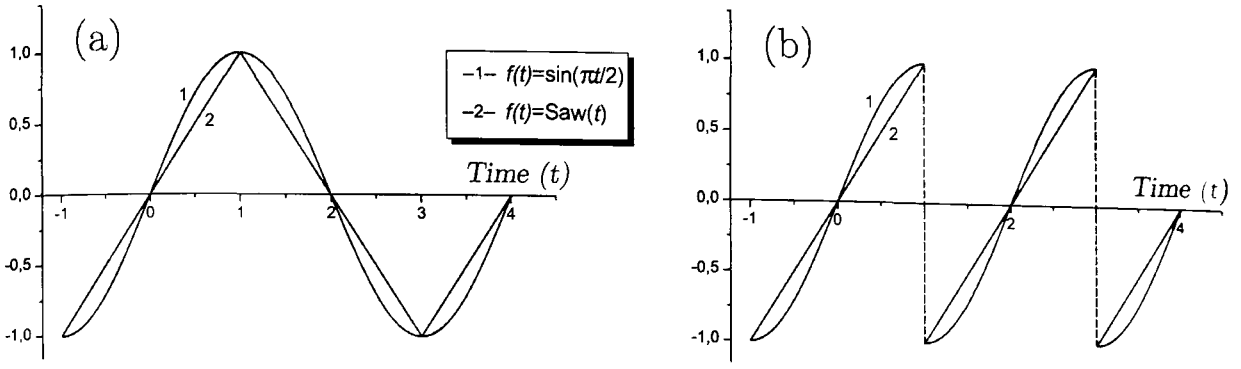


Figure 43. Using signals displayed on figure a) as the basic function $f(t)$ for the synthesis of orthogonal signals will give the same ultimate values a_i as using of signals displayed on figure b), corresponding directly to the mathematical description of the device.

generated signals e.g. 8th order Legendre polynomial). The frequency was restricted to avoid the dynamical behavior of elements such as finite slew rate as well as signals phase shift in OP-amp and multipliers.

It should be noted that the sinusoidal signal and a saw-wave signal (see Fig. 43a) (but not the signals shown in Fig. 43b) were used in the experiments as a basic function $f(t)$ for synthesis of the orthogonal bases shown in Fig. 41. (It is easy to show that in both cases of $f(t)$ defined as shown in Fig. 43a and Fig. 43b, the coefficients a_i that are established after the process of adaptation have the same values).

The signals shown in Fig. 41 from a) to e) have been synthesised experimentally.

The inner product coefficients $C_{ij} \equiv \int_{-1}^{1} F_i(t)F_j(t)dt$ (where i and j are some integer numbers from 0 to 7) for the Smoothed Cosine basis (see Fig. 41d and Fig. 44) synthesised by the experimental breadboard (the data shown in the Fig. 44 were imported from an oscilloscope) are shown in Table 2. It can be seen that the degree of mutual signals' orthogonality is decreasing with increasing degree of the basis F_i. This phenomenon is connected with the fact that nonidealities of elements from previous cascades of the Synthesiser are accumulating and influence the subsequent cascades.[3]

It should be noted here that after the adapting phase (as a result of which the coefficients a_i reach the stationary values) the output signals of the Synthesiser $F_i(t)$ are some stationary nonlinear functions of input signals $f(t)$ and $g(t)$ (due to there not being any time or frequency dependent elements in *feedforward* circuits (such as e.g.

[3] Taking into consideration normalising coefficients of functions, we will see that the nonidealities as well as a noise of previous cascades of the system are not only affecting subsequent cascades, but that this influence increases with the cascade number.

Table 2. The inner product matrix of the synthesised Smoothed Cosine functions' Basis shown in Fig. 44.

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
F_0	3.3692	-0.0028	-0.0062	-0.0101	-0.0199	-0.0015	-0.0037	-0.0139
F_1	-0.0028	3.5898	0.0061	0.0060	0.0097	0.0121	0.0148	0.0040
F_2	-0.0062	0.0061	3.5791	0.0209	0.0072	0.0202	-0.0017	0.0094
F_3	-0.0101	0.0060	0.0209	3.8883	0.0340	0.0137	0.0357	0.0058
F_4	-0.0199	0.0097	0.0072	0.0340	3.7625	0.0522	-0.0001	0.0129
F_5	-0.0015	0.0121	0.0202	0.0137	0.0522	3.7105	0.0608	-0.0146
F_6	-0.0037	0.0148	-0.0017	0.0357	-0.0001	0.0608	3.4168	-0.0325
F_7	-0.0139	0.0040	0.0094	0.0058	0.0129	-0.0146	-0.0325	3.5902

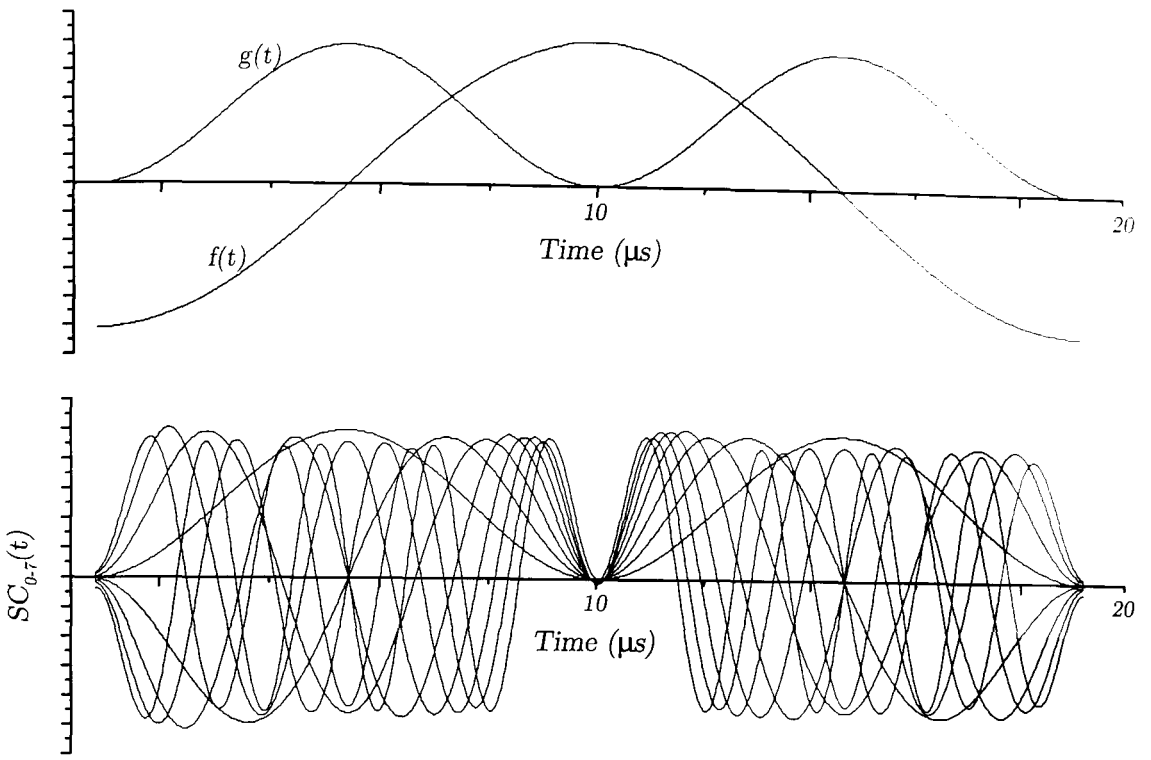


Figure 44. Smoothed Cosine functions generated by the breadboard of the analogue Synthesiser of orthogonal signals. $f(t)$ and $g(t)$ are input signals; SC_{0-7} are output signals.

filters, integrators, etc.), but only analogue multipliers and summators, see Fig. 39). Therefore if at some moment we will change the period of applying signals $f(t)$ and $g(t)$ i.e. if we will change signals $f(t)$ and $g(t)$ respectively to $f(at)$ and $g(at)$ where a is some constant, we will instantly have signals $F_i(at)$ on the outputs of the Synthesiser and these signals will be stationary during the time.

In the future researches we are going to concentrate our attention on problems of stability and speed of convergence of the Synthesiser, whereas in this work we just have done a general description of principles of functioning as well as a description of the proof-of-concept breadboard function.

3.2.6 Conclusions

As a result of this work it is possible to make several conclusions:

- It is possible to produce on the basis of analogue elements a plurality of mutually orthogonal signals such as Legendre Polynomials, Chebyshev Polynomials, Cosine Basis of Functions, Smoothed Cosine Basis, etc.

- The recurrent equations (3.30), where a_i are defined by integral equations (3.31) allow to produce a basis of orthogonal functions $F_i(x)$ if the functions $f(x)$ and $g(x)$ applied at the inputs of the synthesiser are respectively odd and even
- The condition Eq.(3.31 could be realised by means of the simple feedback shown in the Fig. 39(a).
- The process of the synthesis of orthogonal functions is fast and very stable.
- In the case of high precision and high orthogonality requirements, a scheme of an Analogue Synthesiser of orthogonal signals must include some additional units for offset zeroing and normalising in addition to the scheme shown in Fig. 39.
- In the case not the highest requirements of precision and orthogonality it is possible to create the Synthesiser with *not-adapting* but with *pre-set* (in analogue or digital memory) coefficients a_i. In this case the Synthesiser will be a little simpler and will not demand a special phase of adaptation but will be less precise as well as more sensitive to temperature variation and to degradation of elements over time.
- The maximal frequencies of function of the Synthesiser implemented on the basis of up-to-date analogue VLSI technology working are supposed to be several hundreds megahertz. But in addition to the problem of off-sets compensation in a high frequency range, the problem of an inter-cascade delay of signals must be solved.

Such a Synthesiser of orthogonal signals should be useful in telecommunication systems. For example - Smoothed Cosine functions (Fig. 41d) as well as function shown in Fig. 41e are well-localised in both time and frequency and may be used for example for megahertz range modem creation, as well as in code division multiple access (CDMA) digital wireless communication systems (combining binary Walsh functions and orthonormal functions localised in time allows us to create different complex bases with a sufficient number of well-localised in frequency and mutually orthogonal functions).

The Synthesiser may be useful also for creation of fast-training approximators or interpolators of one or multi-variable functions (e.g. for an analogue implementation of Chebyshev polynomials-based (CPB) neural networks that were theoretically described

in [43]). An analogue device for the fast functions' approximation on the basis of Legendre polynomials is described in [44].

It is possible also to use devices based on orthogonal functions as an alternative to Volterra polynomial neural networks (or filters) (described e.g. in [45]). Orthogonal function - based devices will much faster in adaptive mode in comparison with conventional Volterra's networks.

3.3 Suppressor of acousto-optic nonlinear distortions

As mentioned earlier, adaptive polynomial systems are most efficient for approximation of weakly nonlinear functions. Thus the problem of nonlinear distortion compensation is one of the widest areas of application for adaptive polynomial systems. Such systems normally must be not only highly linear, but also - highly precise. That's why the adaptivity of the system could be crucial since the precision characteristics of analogue elements are very sensitive to temperature as well as to long-term degradation.

Analogue polynomial adaptive systems could find application in cases of high frequency systems. An example of such a system is the Volterra filter (see [46]). Such filters could be used for modeling of nonlinear dynamic systems as well as for nonlinear distortion compensation (e.g. in powerful amplifiers for radio stations).

Let us consider here the promising application of adaptive polynomial analogue system for the suppression of intermodulations in Acousto-optical (AO) systems. AO systems are widely used for information processing (AO spectrum analysers of RF-signals as well as AO adjustable filters higher than 100MHz bandwidth, etc.) as well as for laser light manipulation (AO scanners, deflectors, modulators). One of the main parameters of such systems is the dynamic range.

The typical dependence of the efficiency of AO interaction I_a as a function of strain u is shown in Figure 45. In this case the intensity of the incident light is $I_0 = 1$, the level of noise is 10^{-8}. The dynamic range in this case will be the maximal difference between the curve of signal intensity I_a and the curve of intermodulations I_{abc}, which relates to three-tone intermodulations of 3rd order.

The dashed curves relate to a normal (non compensated) AO cell. We can see that the suppression of AO intermodulations could considerably (up to two orders of magnitude) increase the dynamic range of AO systems.

The intermodulation effects in AO systems are described by others in [47]. In [48] and [49] we suggested a new technique of AO intermodulations suppression. This technique is based on the strong nonlinearity of the photoelasticity effect in resonant materials. This allows us to create a new material (based on mixture of non-resonant photoelastic material containing resonant non-linear impurities), which, if used for an AO cell creation, will be free of intermodulations of 3rd order. This will allow us to increase the dynamic range by more than one order of magnitude. In spite of the

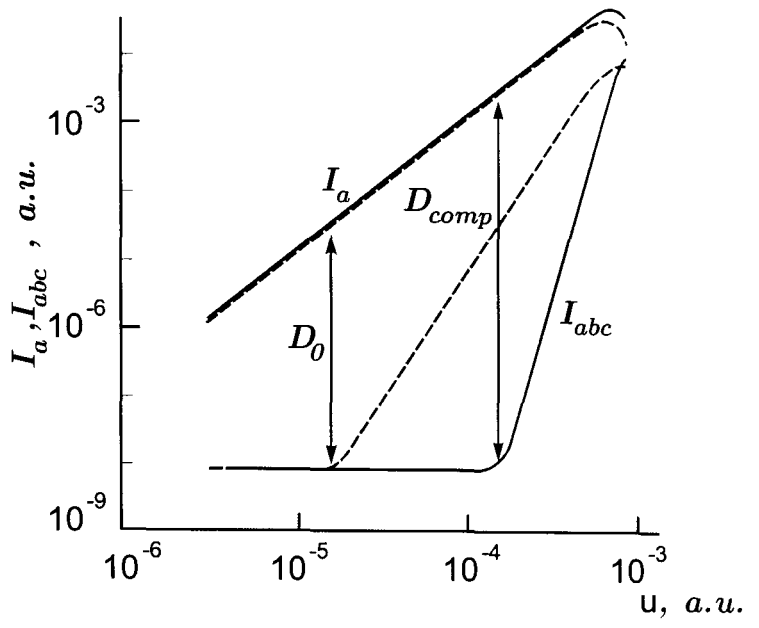


Figure 45. Typical dependence of the efficiency of AO interaction I_a and of the intensity of intermodulations I_{abc} as a function of strain u. It is possible to see that the dynamic range D is about two orders of magnitude higher in the case of suppressed intermodulations (solid lines).

importance of the parameter - dynamic range - it is not a trivial problem to synthesise a new material with appropriate properties. This problem is not solved yet.

On the other hand the same goal (the suppression of nonlinear distortions of 3rd and higher orders) could be reached by means of electrical signal modification. Indeed, by means of bringing pre-distortions into the electrical signal, it is possible to suppress the intermodulations.

Let us consider how this could be done in the case of "thin" AO gratings (when $h \ll \frac{k}{K^2} \operatorname{Re} \sqrt{\varepsilon_0}$), where h is the width of sound beam, k and K are the wave numbers of light and sound waves, respectively). The consideration yields to the amplitude of the propagated light $E \approx E_0 \exp(i\sqrt{\varepsilon(y,t)}kh$.

If $x(t)$ is the input signal, after the 3rd order polynomial modification the electrical signal on the piezoelectrical transducer will be $u(t) = x(t) + \alpha x^2(t) + \varkappa x^3(t)$. The dielectric permittivity $\varepsilon(y,t)$ could be expanded in a series of strain $u(y)$: $\varepsilon(y,t) = \varepsilon_0 + \varepsilon_1 u(y,t)$. Assuming that the input signal is a superposition of sine signals: $x(y,t) = \sum_{m=1}^{N} x_m \cos(K_m y - \Omega_m t + \delta_m)$ (see Figure46), x_m and Ω_m are the amplitude and the frequency of the m^{th} partial sound wave, ε_0 is the dielectric permittivity without sound, ε_1 is the linear photoelastic coefficient, on the output of the AO cell we will have for the amplitude of light: $E(y) = E_0 \exp\left(ikh\sqrt{\varepsilon_0}\right)(1 + Ax + Bx^2 + Cx^3 + ...)$, where

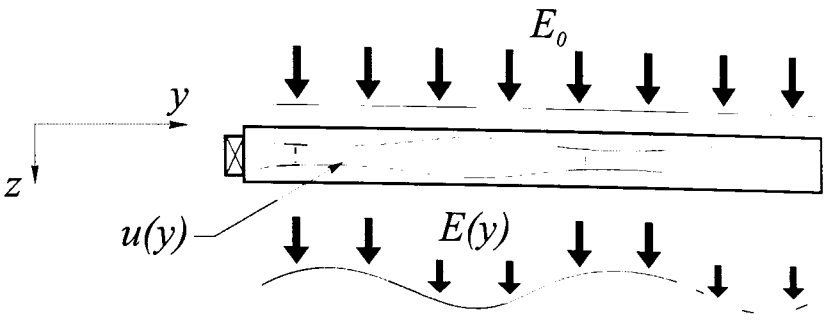


Figure 46. AO interaction in regime of "thin" graiting.

$$A = \beta h \varepsilon_1;\ \beta \equiv \tfrac{ik}{2\sqrt{\varepsilon_0}};\ B = \beta h\left(\alpha\varepsilon_1 - \tfrac{\varepsilon_1^2}{4\varepsilon_0}\right) + \tfrac{\beta^2 h^2}{2}\varepsilon_1^2;$$

$$C = \beta h\left(\varkappa\varepsilon_1 - \frac{\alpha\varepsilon_1^2}{2\varepsilon_0} + \frac{\varepsilon_1^3}{8\varepsilon_0^2}\right) + \beta^2 h^2\left(\alpha\varepsilon_1^2 - \frac{\varepsilon_1^3}{4\varepsilon_0}\right) + \frac{\beta^3 h^3}{6}\varepsilon_1^3. \qquad (35)$$

The coefficients A, B and C correspond to 1st, 2nd and 3rd order AO processes (distinguished by the numbers of diffraction orders). Following the Floke theorem of the superposition (see [48]) we can obtain for the related E components of diffraction orders:

$$E_{(a)} = \frac{A}{2}E_0 x_a \exp\left(i\left(kh\sqrt{\varepsilon_0} + \delta_a\right)\right)$$

$$E_{(a-b)} = \frac{B}{2}E_0 x_a x_b \exp\left(i\left(kh\sqrt{\varepsilon_0} + \delta_a - \delta_b\right)\right)$$

$$E_{(a-b+c)} = \frac{3C}{4}E_0 x_a x_b x_c \exp\left(i\left(kh\sqrt{\varepsilon_0} + \delta_a - \delta_b + \delta_c\right)\right)$$

$$E_{(2a-b)} = \frac{3C}{8}E_0 x_a x_b x_c \exp\left(i\left(kh\sqrt{\varepsilon_0} + \delta_a - \delta_b + \delta_c\right)\right).$$

Thus, both the 3-tone ($E_{(a-b+c)}$) and 2-tone ($E_{(2a-b)}$) intermodulations of third order are proportional to the same coefficient C, which is defined by the Eq.(3.35). The zeroing of this coefficient will suppress **all** the intermodulations of 3rd order, thus will increase the dynamic range by more than one order of magnitude.

In Figure 45 the solid lines are related to AO systems with suppressed intermodulations of 3rd order.

The coefficient C, as we can see from Eq.(3.35), could be zeroed by selection of proper parameters α and/or $\varkappa$. The principle of the compensator function is shown in Figure 47. The input signal goes through a nonlinear (polynomial) adaptive unit which adds the non-linear pre-distortions in order to suppress the intermodulations.

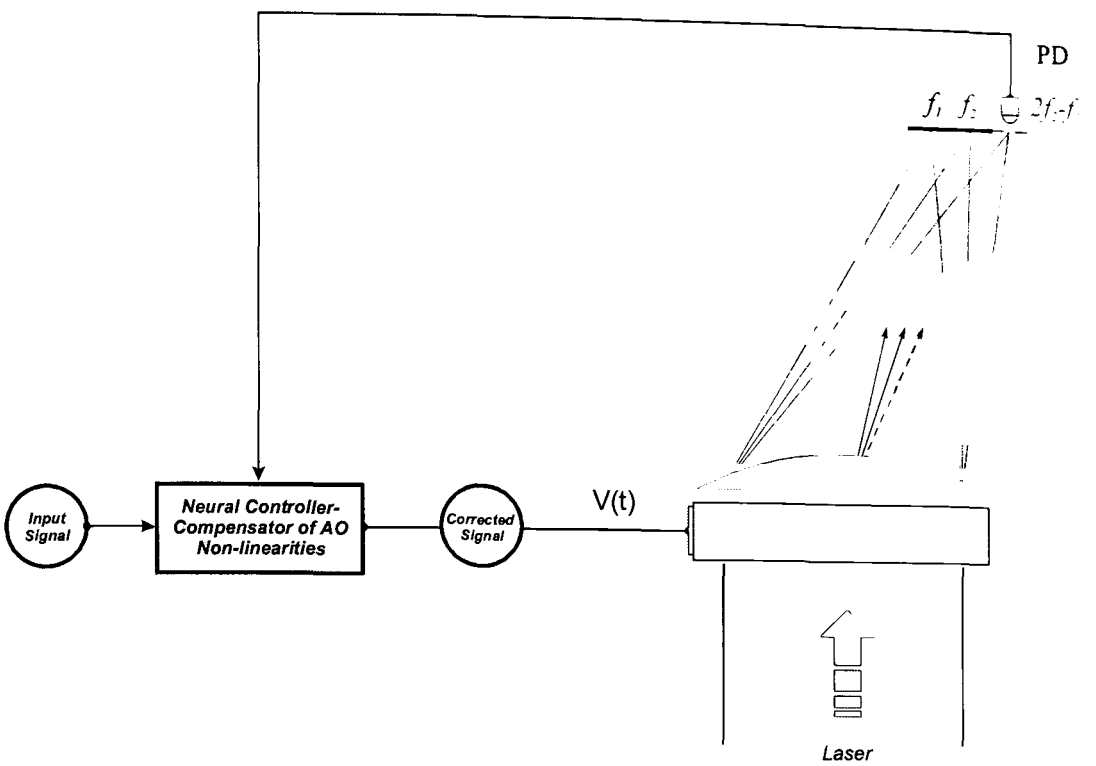


Figure 47. This scheme shows the principle of function of the adaptive polynomial unit, suppressing the AO nonlinearities of 3rd and higher order.

To compensate for 3rd order nonlinear distortions (AO intermodulations) on the self-calibration stage of the AO device, the test signal, consisting of two sine signals $x = \sin(f_1 t) + \sin(f_2 t)$ is applied to the input of the system (input signal in Figure 47). The intermodulation signal with frequency $(2f_2 - f_1)$, result in the nonlinearity of AO interaction, will be spatially resolvable (see Figure 47). This parameter could be used as a "parameter of quality" for adjustment of the polynomial transfer function (see Figure 48). *By zeroing of this "parameter of quality" we will suppress* all *the 3rd order nonlinearities of the AO cell.*

It should be noted that the suppression of AO intermodulation in practice will not give as good result as in theory. The reason is that there are other contributions to the nonlinearities of the systems as well as the nonlinearity of AO interaction. Every amplifier, modulator, as well as the piezo-electrical transducer are contributing both to the nonlinearity and to phase distortions of AO systems. To suppress not only the nonlinear distortions but also phase distortions, the scheme shown in Figure 49 was suggested.

The perturbation-based system, used in the system, implements the following procedure: subsequently one after another each degree of freedom W_i is

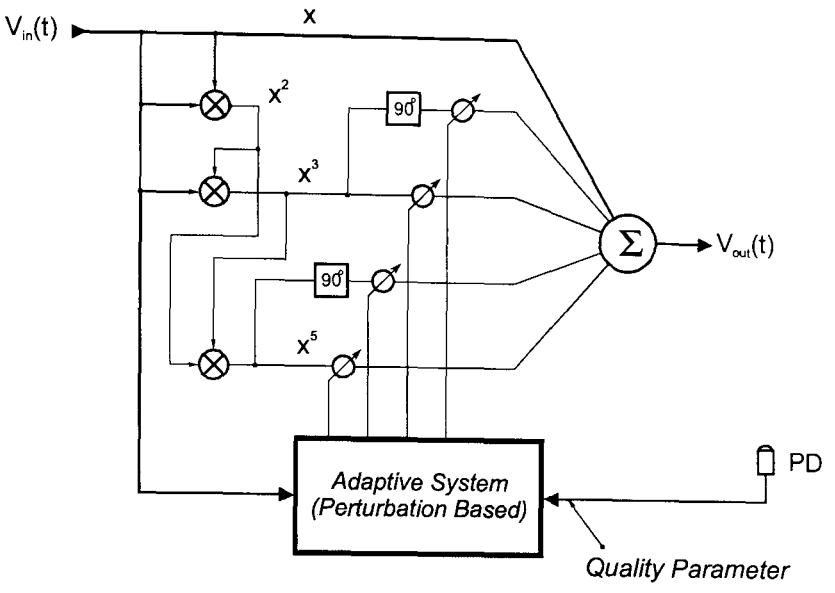


Figure 48. Scheme of the adaptive polynomial suppressor of AO nonlinearities.

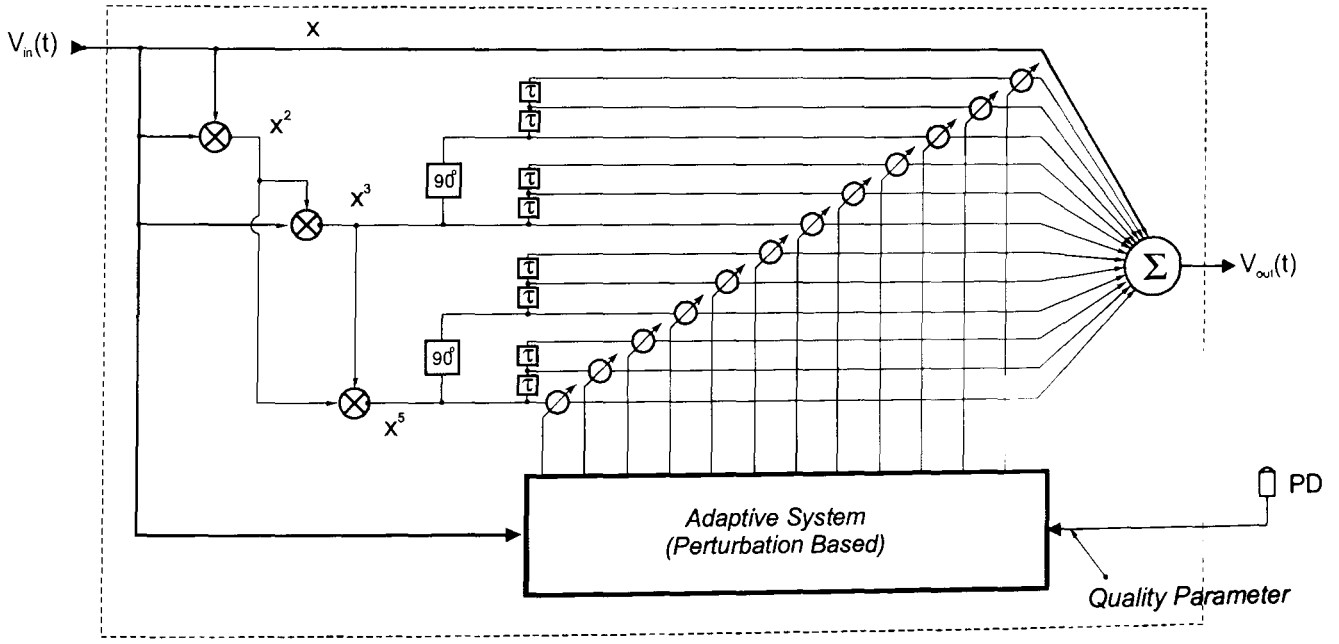


Figure 49. Scheme of the adaptive polynomial suppressor of AO nonlinearities for the case of the phase-distorted signal.

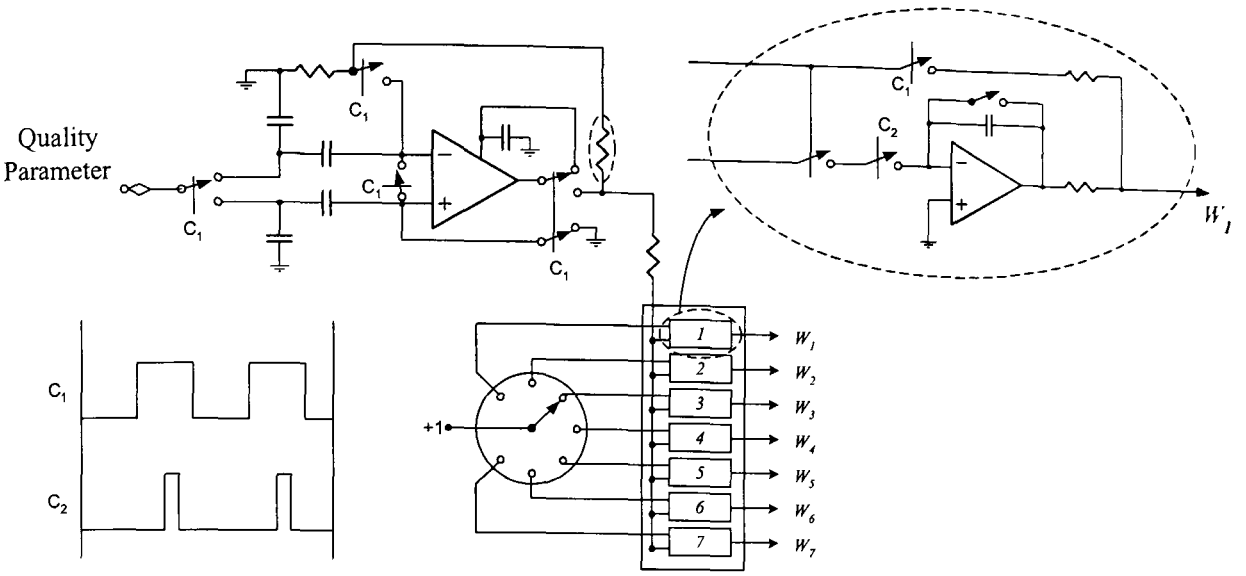


Figure 50. An example of implementation of the perturbation-based adaptive unit. This scheme is optimising the W_i coefficients aiming to minimise (or maximise) the "quality" parameter.

1) getting some small increment: $\widetilde{W}_i(k) = W_i(k-1) + \Delta$;

2) if the "parameter of quality" improves, the new value of the current degree of freedom is fixed: $W_i(k) = \widetilde{W}_i(k)$;

if not - the current degree of freedom is changed to $W_i(k) = \widetilde{W}_i(k) - 2\Delta$.

An example of implementation of the perturbation-based adaptive unit is shown in Figure 50. This scheme optimises the W_i coefficients aiming to minimise the "quality" parameter. The of AO distortion Suppressor should be implemented on the basis of analogue technology because the sound normally used for AO interaction is very high frequency (hundreds of megahertz) (thus the feedforward part of the system should be analogue). Besides, the adaptive unit of the system also should be analogue to maximise its precision (since digitizing inevitably will affect the precision). On the other hand the results of such optimisation could be stored into a digital memory.

3.4 A new technique for indirect identification of extracellular electrode position

It is a well known fact that the electrical signal, taken by extracellular recording (ECR) technique from a neural cell, strongly depends on the position of the electrode in relation to the cell body (see Figure 51).

The signal is not only becoming weacker with increasing distance between the electrode and the cell, but it also changes its shape, especially when the electrode is moved along the axon. This phenomenon could be explained by the complex dynamics of diffusion of different carriers of electrical charge, which are ejecting into the medium during the neural cell firing. The delay of signal propagation along the axon is also contributing to the complex shape of the signal.

Here we are describing the artificial neural network (ANN) based model of the system: neural cell + electrolytic medium. The proposed technique is based on experimental data (where extracellular electrical signals have been recorded in different positions with respect to the neuron). All data used at the stage of the model creation supposed to be noisy.

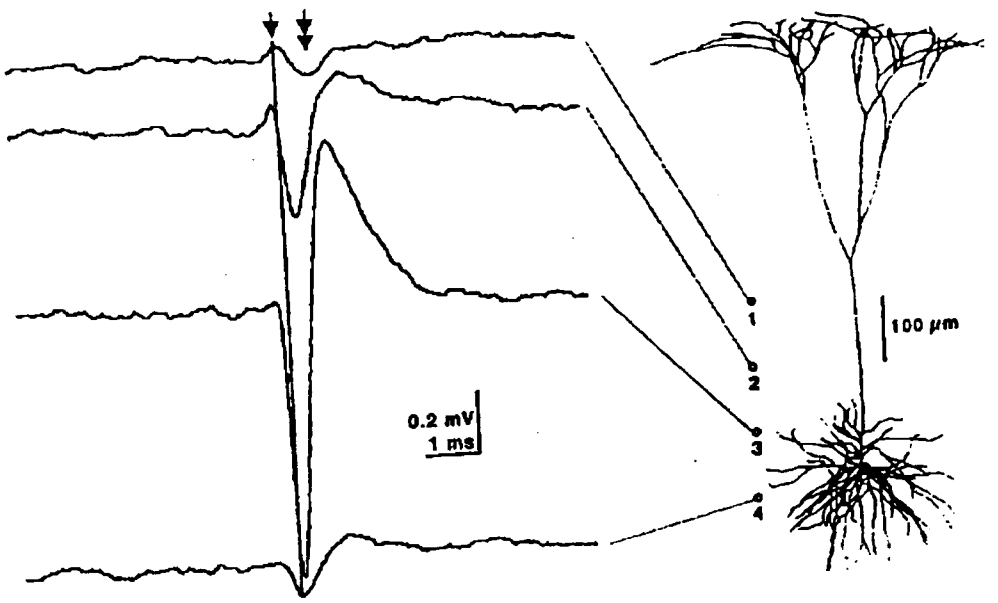


Figure 51. Shape of the neural signal recorded by extracellular neural measurements strongly depends on the position of the electrode.

Once the model has been created the signal related to some new position of electrode can be predicted. This also lets us identify the position of the electrode on the basis of analysis of the shape of the neural signal.

We can not see at the moment how the results of the work could be directly used in practice because the model strongly depends on the geometry of the particular system (neuron+medium). On the other hand the proposed technique (and mathematical apparatus) could be part of an advanced procedure of Blind Signal Separation (BSS) of neural signals received by one electrode from two or more neurons.

Now we will discuss only the task of definition of the electrode position on the basis of analysis of signal shape (the reversed task of prediction of signal shape in some particular place will not be considered).

3.4.1 Formal task statement

Let us suggest that some amount of neural signals d_i were taken (recorded) from an area around a neuron at positions of electrodes x_i, y_i (for the sake of simplicity let's consider 2D geometry). Here i is the number of measurements ($i < N$).

Our task will be the creation of a signal processing procedure which will let us define the position of the electrode (x_N, y_N) by means of analysis of the shape of signal d_N taken during a **new** measurement.

The setup of the proposed system is shown in Figure 52.

Initially we have the electrode signal as a function of time. The task of the **pre-processing** system is:

1) identification of the neural signal;

2) noise filtration;

3) separation of critical parameters of signal (information "compression");

On the output of the pre-processing unit we will have a set of discrete parameters W_i , which are the coefficients of decomposition of the neural firing signal.

The next stage is the **Neural Network based analysis** of decomposition coefficients W_i. The task of the neural network is to establish the implicit relation between the decomposition coefficients and the output parameters: position in space in relation to neural cell (x, y), and the time delay of Δt of the received signal $d(t)$ in relation to spike detector response which occur at the moment t_0. The task is complicated by the

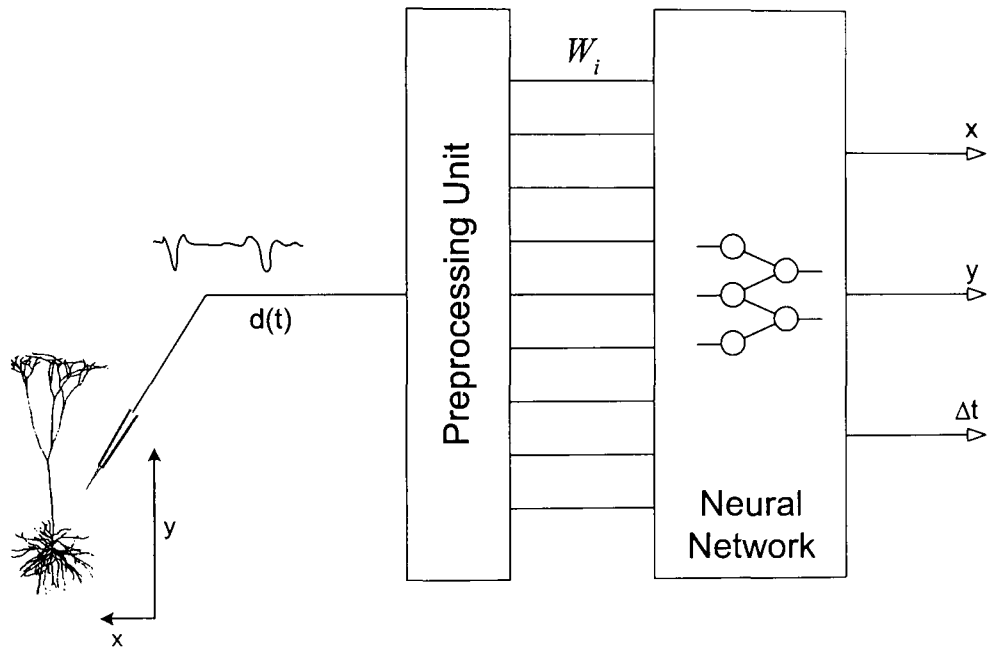


Figure 52. Setup of the electrode's position identification system

fact that the received signal $d(t)$ is noisy and therefore there is a big ambiguity (i.e. error) in time when the spike-detector could respond to firing of the neuron.

3.4.2 Pre-processing System

General Principles The scheme of the pre-processing system is shown in Figure 53. The pre-processing system performs the decomposition of the neuronal signal into series of ortho-normal functions F_i:

$$d(t) \approx \sum W_i \cdot F_i(t) \tag{36}$$

where

$$W_i = \int_{t_0}^{t_1} d(t) \cdot F_i(t) dt.$$

The next stage of signal processing (the ANN based identification system) will be dealing only with the coefficients of decomposition W_i. We don't need to approximate precisely the input signal $d(t)$, therefore we could truncate the series and use just sev-

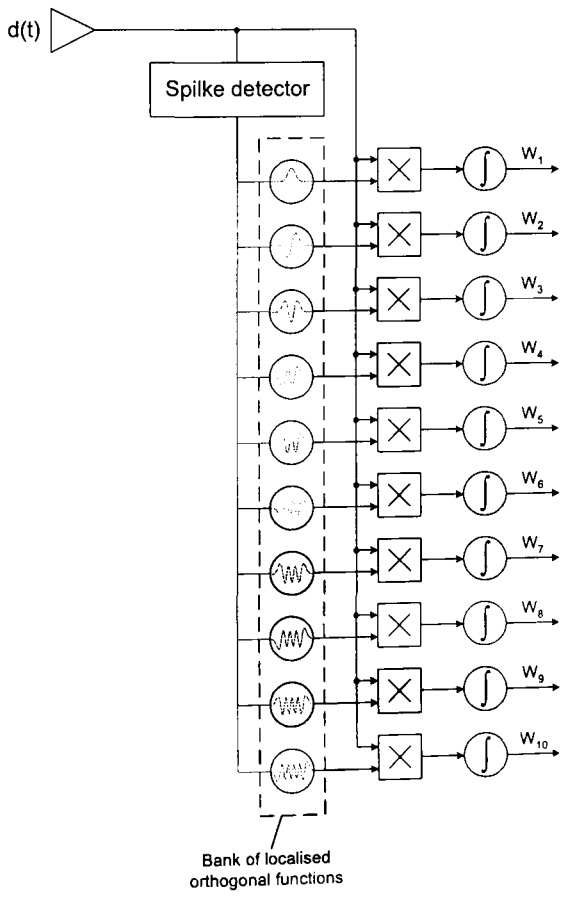


Figure 53. Scheme of the pre-processing unit

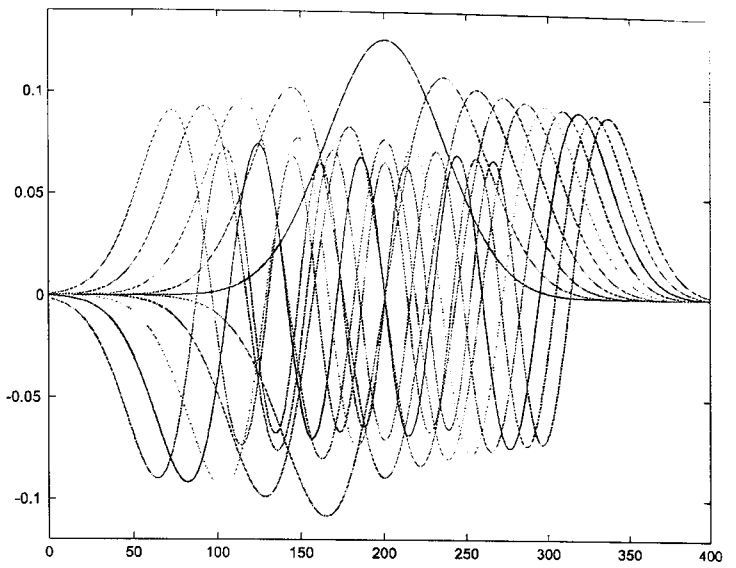


Figure 54. Localised orthogonal functions used for the pre-processing of neuronal signals.

eral coefficients of the decomposition. This will allow us to decrease the information characterising the spike. On the other hand, the proper selection of an ortho-normal basis of functions will let us minimise the influence of noise of the input signal on the information processing.

The criteria which should be fulfilled at the stage of basis function selection are:

- basis functions should be ortho-normal;
- basis functions should be finite;
- the power distribution of the basis functions $F_i^2(t)$ should conform to a typical shape of the neuronal signal $\overline{d(t)}$.

Implementation The spike detector is reacting to the neuronal signal's emergence and launches the orthogonal signal $F_i(t)$ generation. In Figure 54 the orthogonal functions that were used for neuronal signal approximation are shown. To synthesise this basis of functions the technique described in section 3.2 was used. This technique lets to synthesise a variety of orthogonal bases, so that we can select the appropriate basis in accordance with the above-mentioned criteria. In Figure 55 the scheme of the Synthesiser of Orthogonal Functions is shown. To synthesise signals shown in Figure 54, the Gauss function and the Sine function were applied to the inputs $g(t)$ and $f(t)$ respectively.

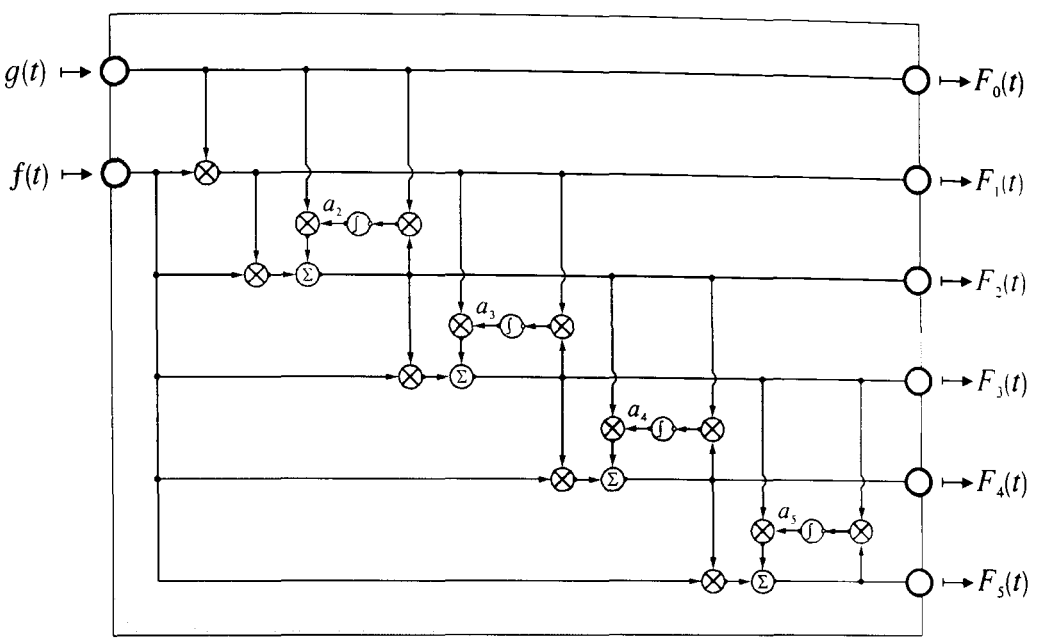
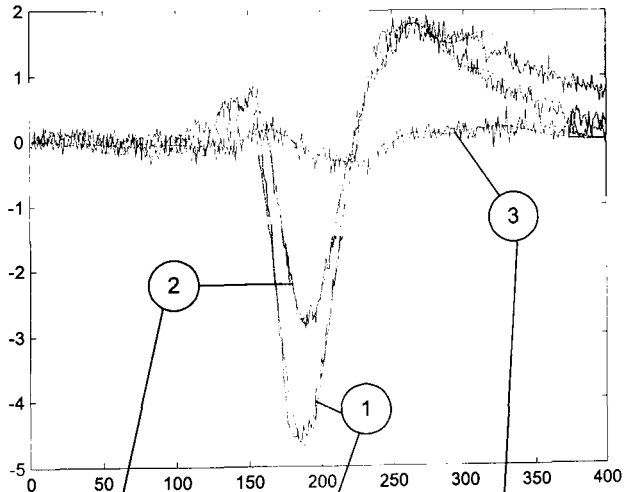


Figure 55. Scheme of the synthesiser of orthogonal functions



W_1 - W_{10}	W_1 - W_{10}	W_1 - W_{10}
-9.62	-8.08	-1.23
8.26	26.15	-1.62
10.83	18.46	0.25
-0.95	-1.43	1.94
0.34	3.13	0.26
4.28	10.58	0.42
1.54	3.29	0.61
-0.31	0.29	0.63
1.86	3.96	0.26
1.51	4.26	0.78

Figure 56. Recorded (noisy) data $d(t)$ and approximation curves (smooth). Here W_i are the decomposition coefficients, characterising the approximating functions in accordance with Eq.(3.36).

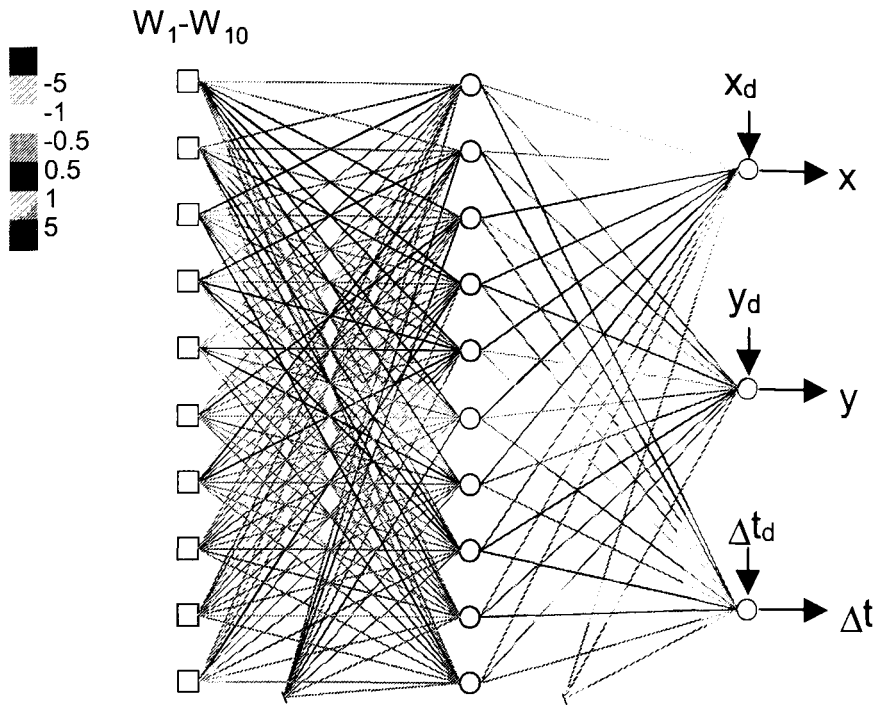


Figure 57. Scheme of the neural network used for electrode position identification.

Examples of approximated functions $d(t)$ are shown in Figure 56. Here the recorded signals are noisy whereas the approximating signals are smooth. The corresponding decomposition coefficients $W_1 - W_{10}$ are also shown in Figure 56.

3.4.3 Neural Network

The Artificial Neural Network (ANN) unit of the signal processing system is dedicated to establish the interrelation between the decomposition coefficients $W_1 - W_{10}$ on its input and (1) the location of electrode in space (x and y) and (2) the time delay Δt of the centre of spike in relation to the moment of the spike detector reaction t_0 on its output (see Figure 57).

To establish this interrelation the set of N measurements $d^{(n)}(t)$ (where n could be from 1 to N) taken at locations $(x^{(n)}, y^{(n)})$ is used. The corresponding decomposition coefficients $W_i^{(n)}$ are applied to inputs of the ANN, whereas the expected outputs x_d, y_d and Δt_d are applied to the output of the ANN.

The task of the neural network during the learning stage is to get the proper response ($x^{(n)} \simeq x_d^{(n)}$, $y^{(n)} \simeq y_d^{(n)}$, $\Delta t^{(n)} \simeq \Delta t_d^{(n)}$) to input signals $W_i^{(n)}$.

The dynamics of learning of the ANN is shown in Figure 58.

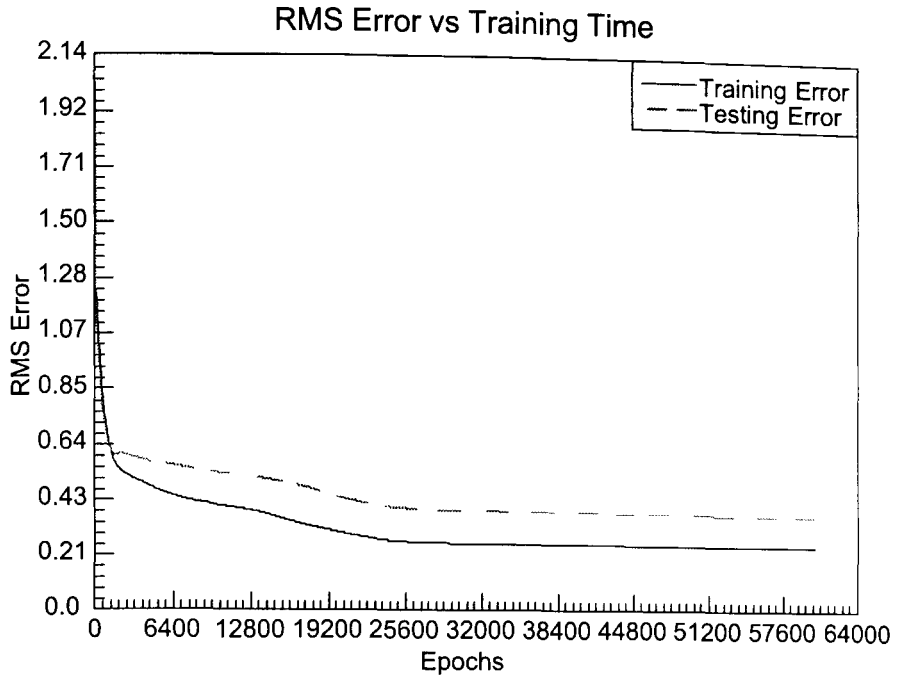


Figure 58. Learning curve $E(t)$ - solid line and the testing curve $E_{test}(t)$ - dashed line.

Once the interrelation is established (after the learning stage), we can assume that if we apply a new set of M measurements $d^{(m)}(t)$ after pre-processing and ANN on the output of neural network we will have the predicted locations $(x^{(m)}, y^{(m)})$ of the electrode. In Figure 59 the white circles symbolise the predicted locations of the electrode. The dark circles are the actual positions of the electrode. The lines are connecting the predicted and actual positions of the electrode. Some of the dark circles are connected with several white circles, which means that there were several measurements with different time delay (Δt) but with the same position of the electrode $(x_d^{(m)}, y_d^{(m)})$.

In Figure 58 the solid line is the "learning curve" which is the dependence of the Root Mean Square (RMS) Error E as a function of the learning time, where

$$E \equiv \frac{1}{N} \sum_{n=1}^{N} \sqrt{\left(x^{(n)} - x_d^{(n)}\right)^2 + \left(y^{(n)} - y_d^{(n)}\right)^2 + \left(\Delta t^{(n)} - \Delta t_d^{(n)}\right)^2}.$$

The dashed line in Figure 58 is the dependence of the Testing Error E_{test} as a function of learning time, where

$$E_{test} \equiv \frac{1}{M} \sum_{m=1}^{M} \sqrt{\left(x^{(m)} - x_d^{(m)}\right)^2 + \left(y^{(m)} - y_d^{(m)}\right)^2 + \left(\Delta t^{(m)} - \Delta t_d^{(m)}\right)^2}.$$

It could be seen that the testing curve is following the learning curve although only the N-set of measurements is involved in the procedure of learning. Nevertheless the interrelation established during ANN learning gives the prediction of electrode position

for the M-set of measurements. The errors of both N-set of measurements E and test measurements E_{test} are caused by noise of measurements as well as by the inaccuracy of the model.

We have tried several neural architectures with different number of layers and different number of neurons in each layer in order to define the best parameters. The activation function of the hidden neurons was sigmoid. The output neurons had linear activation function. In Figure 57 the optimal architecture is shown. It has 10 inputs, 10 hidden neurons and 3 outputs. If we increase the number of layers it will diminish the error E but will increase the test error E_{test} (the so called effect of overlearning). If we increase the number of neurons in the hidden layer of 2-layer ANN, it will not considerably increase the accuracy of the interrelation but will overcomplicate the ANN.

It should be stressed that the parameter of quality of the ANN in this case is not the minimum of E in result of learning, but the minimum of E_{test}. It is very important that the system is not only approximating the learning data, but also providing an adequate (smooth) multidimensional interpolation of implicit functions. Such kind of interpolation seems to be appropriate in this case, because the dependence of electrical potential as a function of position of the electrode is cased by diffusion of carriers of charge, and therefore could be expected to be smooth.

It should be noted here that we are **not** considering the proposed system as a new useful instrument for neurophysiologists (since the model of the system we were using is oversimplified and it is difficult to imagine a situation when this kind of system could be useful in practice). On the other hand the above problem in our opinion is very attractive for ANN specialists, since this case is a good example where the inefficiency of other approaches, such as analytical approach as well as the standard computational mathematics methods, to describe complex dynamical systems becomes obvious. So, ANN not only gives us new highly efficient computational architectures (suitable for real-time applications), but also a powerful mathematical apparatus for modelling of complex non-linear dynamic systems.

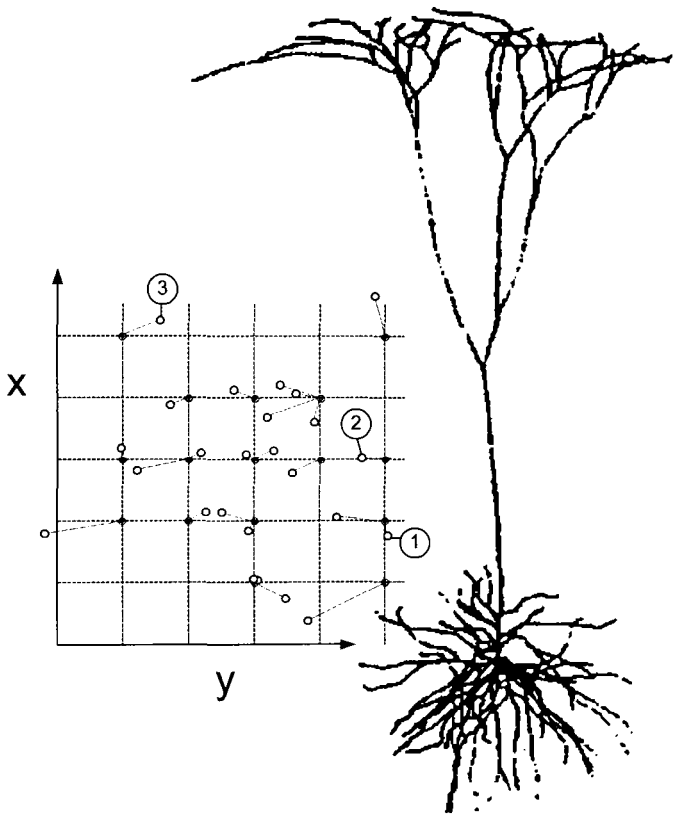


Figure 59. Positions of the electrode: actual (dark circles) and predicted (white circles) by ANN (on the basis of neuronal signal analysis) .

3.5 Real-time analogue video-filter (homomorphic filtering)

It is a well known fact that the eye performs a complex nonlinear transform of an image. The question is however: what exactly is the kind of transform. This question has yet not been answered. Nevertheless there are many algorithms for image transformation including mimicking a human eye and increasing the contrast of an image. Such algorithms are called image enhancement algorithms.

There are different approaches to the problem of image enhancement. The two most traditional approaches are Homomorphic Filtering and Histogram Equalisation (see [38].)

Homomorphic Filtering In this case the algorithm is "equalising" the average luminance across the image. So that if some region is too dark (or too bright) the algorithm is making it brighter (darker). The block-scheme of the Homomorphic Filtering algorithm is shown in Figure 60 and its function is described by $y = \exp(\mathbf{HPF}(\log(x)))$, where x is an input image, y is an output image, **HPF(.)** is an operator of high-pass 2D spatial filtering.

Histogram Equalisation This algorithm analyses the number of pixels with different levels of brightness. On the basis of this statistical information, the algorithm constructs a non-linear transfer function, which makes the distribution of pixels over different levels of brightness more uniform.

Other Algorithms We can see that the Homomorphic Filtering algorithm works in the spatial domain (that is, the algorithm performs a local linear stretch of brightness: if we consider some region, which is smaller than the cut-off spatial frequency of the 2D filter, its brightness after the transform will be $y = a * x$, where a is a constant related to this region, x is an initial brightness). In contrast, the Histogram Equalisation algorithm

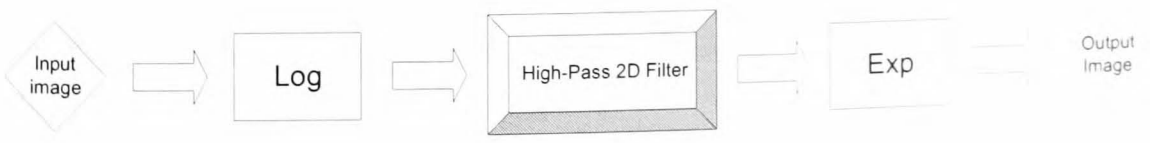


Figure 60. Block-scheme of the Homomorphic Filtering algorithm

works in the brightness domain: $y = \Theta(x)$, where $\Theta(.)$ is some nonlinear function, which normally monotonically increases from 0 to 1 if $\min(x) = 0, \max(x) = 1$.

There are many other algorithms of image enhancement (see e.g. [50]). All of these algorithms are *compressing the dynamic range* of an image. Our eyes are obviously also compressing the dynamic range since the dynamic range of images, perceived by eyes (more than 1000:1) is apparently wider than the dynamic range of nerves, delivering the image-signal from eye to brain (which is about 100:1). The hardware implementations of image enhancement algorithms are mostly digital.

Carver Mead's analogue VLSI Artificial Retina (see section 2.3.1) had made a great impact in the development of science and technology of neuromorphic systems. The artificial retina enhances the contrast of an image as well as compresses the dynamic range of an image. In fact the Silicon Retina [6] implements a simplest version of the homomorphic filtering.

The advantage of analogue implementations of image enhancement algorithms is that they are free of additional noise caused by digitising (which is the initial stage of any digital image processing procedure). Therefore analogue systems of image enhancement might be smaller, less power consuming and cheaper than digital equivalents.

Analogue implementation of homomorphic filtering As far as the Silicon Retina is concerned, its irrationality is quite obvious (as was mentioned in the Introduction), since the picture transmitted by normal video is changing relatively slowly (slower that 50 Hz,) whereas transistors can operate much faster (more than 1 GHz). The real-time analogue image enhancement system suggested in this section much more efficiently uses the potential of transistors. The proposed system performs a slightly modified version of the Homomorphic Filtering (in comparison with the version described above). The block-scheme of the algorithm, which was used for creation of the

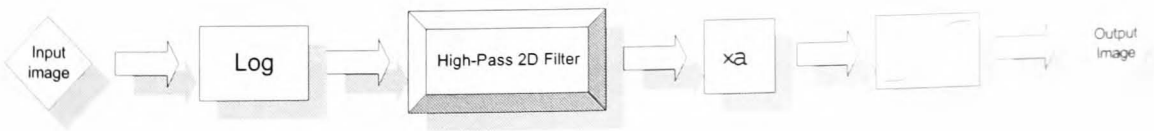


Figure 61. Block-scheme of the modified Homomorphic Filtering algorithm which was used for the analogue image enhancement system.

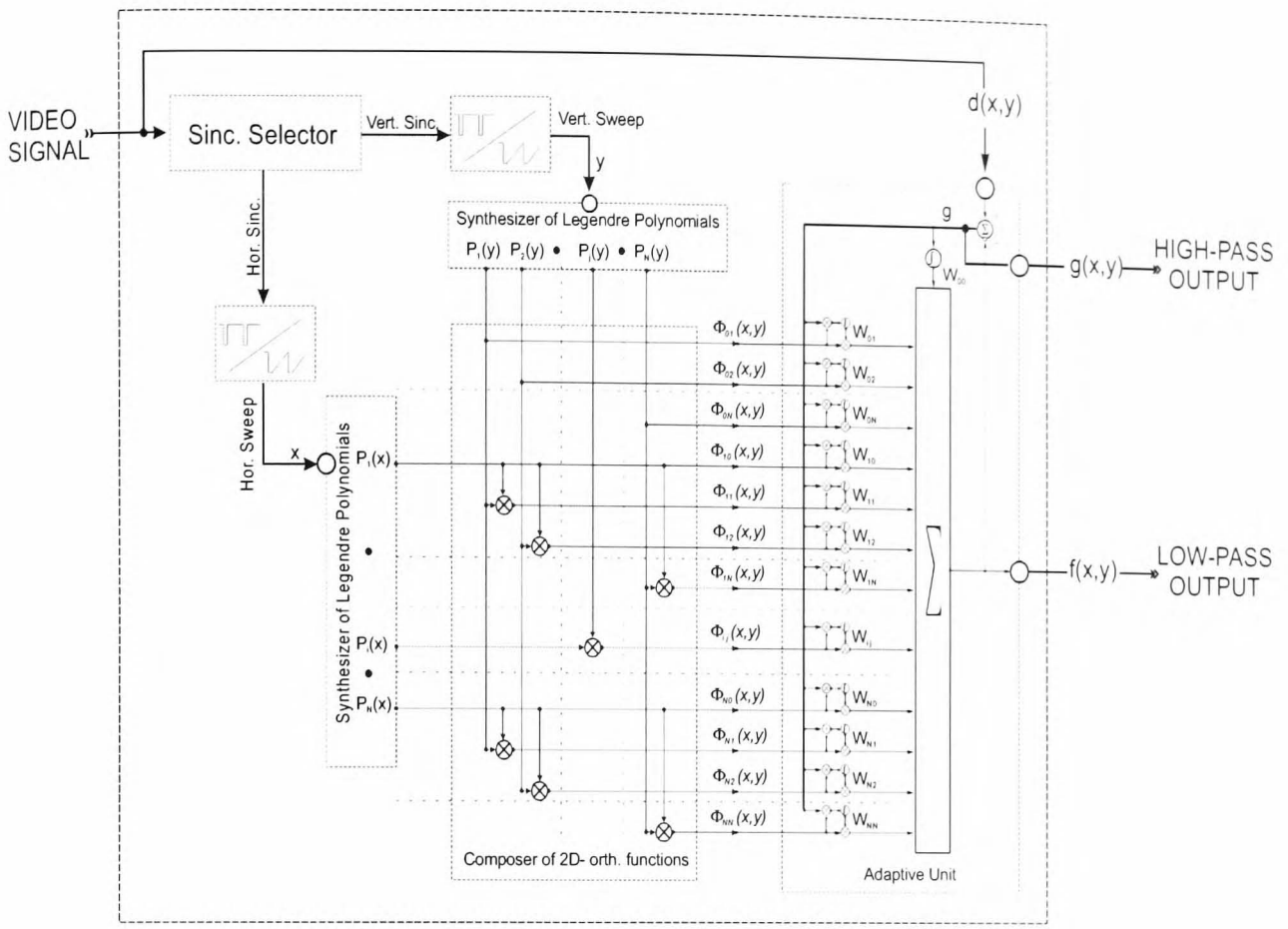


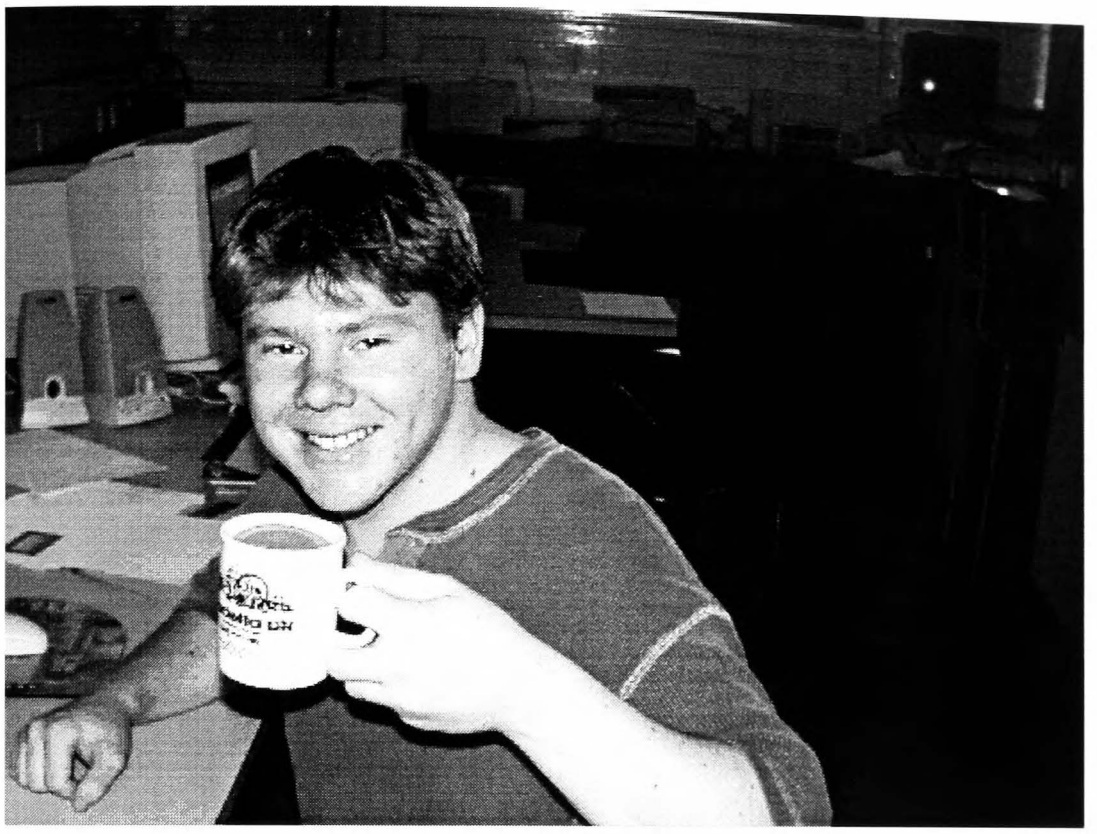
Figure 62. Scheme of analogue 2D spatial low-pass and high-pass video filter.

analogue image enhancement system is shown in Figure 61.The main unit of this system is a high-pass 2D spatial filter. In Figure 62 the electrical scheme of the analogue 2D spatial low-pass and high-pass video filter is shown.

The filter consists of:

1. the sync. selector - the unit extracting the Synchro-Signals from the video signal;
2. the sweep-signal generators, $x(t)$ is a horizontal sweep and $y(t)$ is a vertical sweep;
3. synthesisers of orthogonal functions (Legendre polynomials) $P_{0-N}(x)$, and $P_{0-N}(y)$ (described in section 3.2);
4. composer of 2D orthogonal basis of functions $\Phi_{0-N^2}(x, y)$
5. adaptive unit, which performs the quasi-real-time polynomial 2D approximation of the reference image $d(x, y)$, and thus gives us the low-pass filtered image $f(x, y)$. The subtraction of the low-pass image from the initial image gives us the high-pass 2D spatial filtered image $g(x, y)$.

In Figure 63 the function of Homomorphic Filtering is illustrated (image (a) - before, (b) after filtering). We can see that the dark area of the image become brighter,



(a)

(b)

Figure 63. Example of the homomorphic filtering, where (a) is an input image, (b) is an output image.

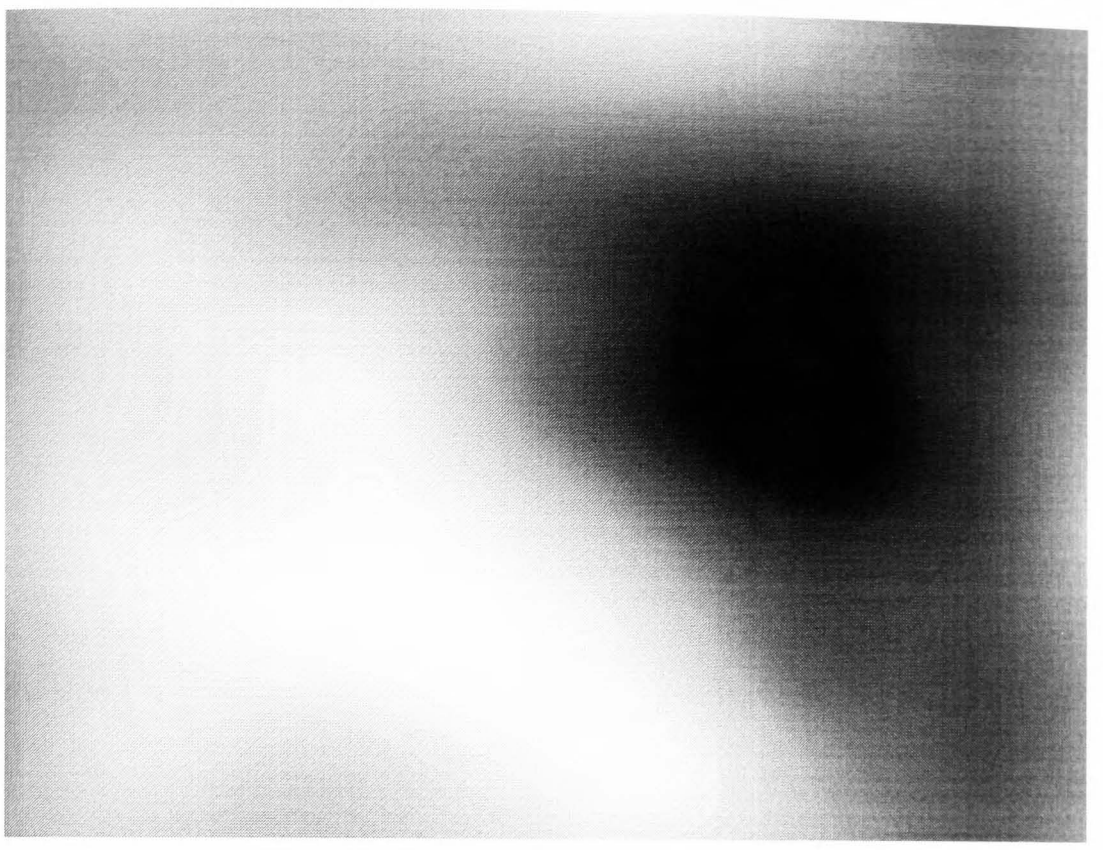


Figure 64. Result of low pass filtering of the initial image.

thus more details have become visible. On the other hand the contrast of the face of the man shown in the picture has decreased. We can see also some artefacts, which are typical for homomorphic filtering: the ear is too bright and there are some other problems in the intermediate area between dark and bright parts of the initial image. In Figure 64 the result of low pass filtering of the input image is shown. It should be noted that very often it could be reasonable to make the "strength" of the algorithm adjustable. This means that the average brightness of different regions of a picture should not be completely equalised, but the brighter parts of the image after the transform should remain brighter than the darker ones. Mathematically this could be realised by the following formula:

$$y = (\alpha - 1)x + \alpha \mathbf{HPF}\left(\log(x)\right),$$

where α is a strength of the transform. In Figure 65 the result of Homomorphic Filtering with the strength $\alpha = 0.5$ is shown. The high/low-pass 2D filter described above could be considered as a quasi-real-time video processing system. In fact it could take several frames for the convergence of the system and establishment of the W_i coefficients.

Figure 65. 50% "mixture" of the input image and low-pass filtered logarithm of the input image.

Since normally an image is changing quite slightly over several frames, the speed of the convergence of the filter shown in Figure 62 is fast enough to follow the changes.

On the other hand a small modification of the system makes the system a real-time image processing device. The changes are:

1. the 2D basis of orthogonal functions should be normalised: $\int_x \int_y \Phi_{ij}^2(x,y)dxdy = \delta_{ij}$, where δ_{ij}.is the Kronecker delta: $\delta_{ij} = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j \end{cases}$,
2. the adaptive unit of the system should contain the sample-and-hold amplifiers.

The modified 2D video-filter is shown in Figure 66.

Such a system transforms the subsequent frame on the basis of information taken from the previous one. Thus the low-pass filtered video signal is delayed by just one frame.

The analogue Homomorphic Filter could be implemented as a small, low power consuming chip and could be used e.g. in security cameras (dealing with nonuniformly illuminated scenes).

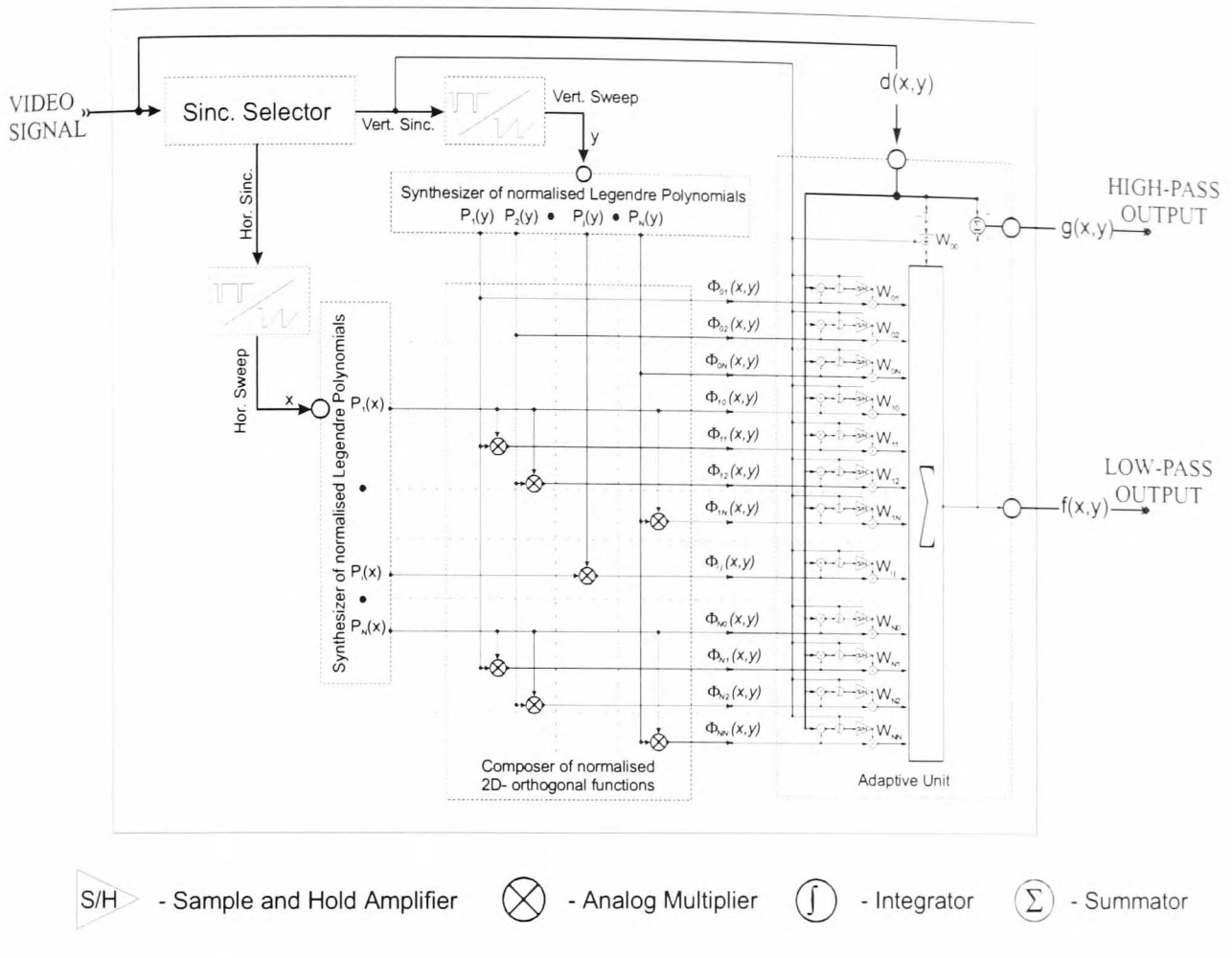


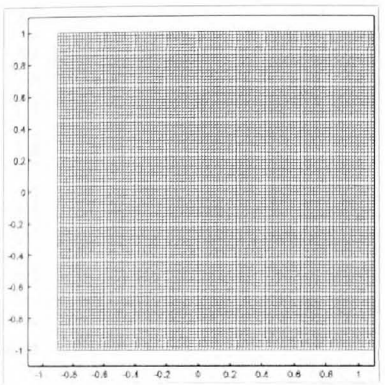
Figure 66. Scheme of "one-step" analogue 2D spatial low-pass and high-pass video filter.

The analogue implementation of low-pass and high-pass 2D filters could be used also as a building block for more complex real-time analogue image enhancement systems.

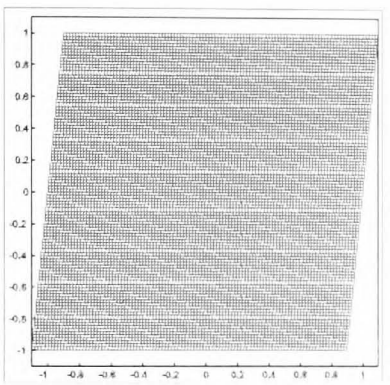
3.6 Adaptive polynomial compensator of geometrical distortions of CRT-monitors

Many imaging systems (in particular cathode-ray tube (CRT) displays) give rise to geometrical distortions of the image. Different mechanisms contribut to the distortions, including the nonlinearities of the electrical circuits as well as peculiarities and imperfections of the CRT construction.

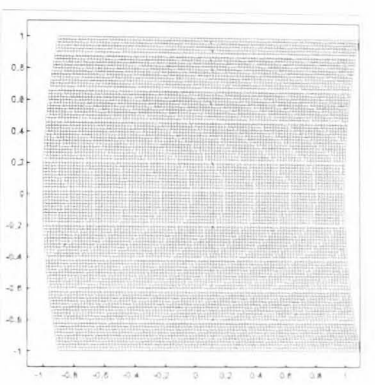
Several kinds of the geometrical distortions are shown in the following table:



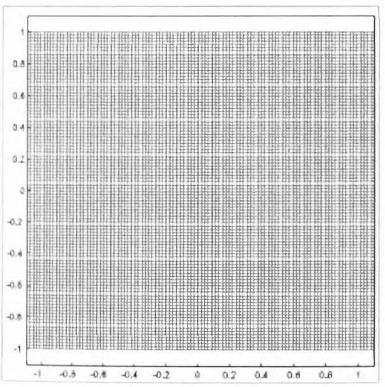
$w_{00} = 0.1$



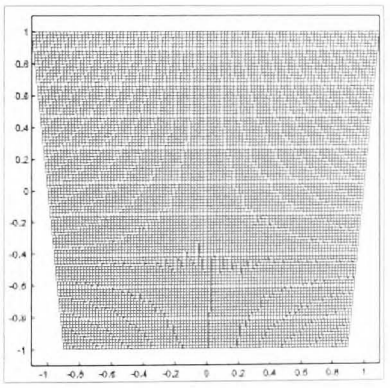
$w_{01} = 0.1$



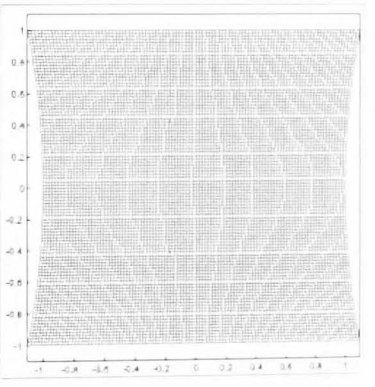
$w_{02} = 0.1$



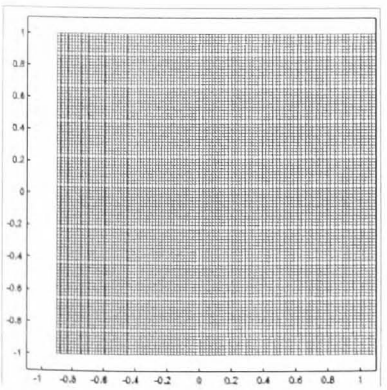
$w_{10} = 1.1$



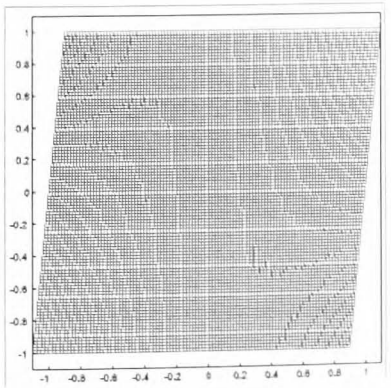
$w_{11} = 0.1$



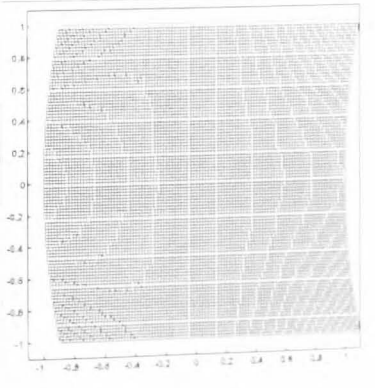
$w_{12} = 0.1$



$w_{20} = 0.1$



$w_{21} = 0.1$



$w_{22} = 0.1$

The example of the geometrically distorted image is shown in Figure 67.

The geometrical distortions are normally weak and hence could be approximated by polynomials:

$$\begin{aligned} x' &= w_{00}^{(x)} + w_{10}^{(x)}x + w_{01}^{(x)}y + w_{11}^{(x)}xy + w_{20}^{(x)}x^2 + w_{21}^{(x)}x^2y \\ &\quad + w_{22}^{(x)}x^2y^2 + w_{12}^{(x)}xy^2 + w_{02}^{(x)}y^2 + .. \end{aligned} \tag{3.37a}$$

$$\begin{aligned} y' &= w_{00}^{(y)} + w_{10}^{(y)}x + w_{01}^{(y)}y + w_{11}^{(y)}xy + w_{20}^{(y)}x^2 + w_{21}^{(y)}x^2y \\ &\quad + w_{22}^{(y)}x^2y^2 + w_{12}^{(y)}xy^2 + w_{02}^{(y)}y^2 + .. \end{aligned} \tag{3.37b}$$

where x and y are the coordinates of an ideal image, e.g. the $(533, 543)th$ pixel relates to $x = 533$ and $y = 543$, x' and y' are coordinates of a distorted image, $w_{ij}^{(x)}$ and $w_{ij}^{(y)}$ are constants describing the nonlinear distortions.

In the above table the different geometrical distortions of x-coordinate: $x' = f(x, y)$, $y' = y$ are shown. Here we assume that $w_{10}^{(x)} = 1$ and all $w_{ij}^{(x)} = 0$, except the coefficients placed below each picture.

Normally geometrical distortions are suppressed at the stage of CRT display (or TV-set) manufacture. Due however to the long-term degradation as well as the temperature sensitivity of analogue elements, residual distortions still could be considerable.

The geometrical distortions (e.g. horizontal) of a display can be visualised by the following technique:

1) a film with a grating, consisting of vertical strips, is placed in front of the display. This mask represents y.

2) the image consisting of vertical strips is displayed on the screen, representing y'.

The result image shown in Figure 68. Here the dark and bright areas are related to the match and mismatch of the strips of grating and the vertical lines pictured by the display. The discrepancy (y' and y) between neighboring dark and bright areas is about 1 mm. The maximal discrepancy between the "ideal" and real image for this particular display is about 4 mm.

For normal computer users nonlinear geometrical distortions of such a small degree is not a problem. However if the user is a computer designer, or in some specific cases when a display is used for measurements of distances, higher precision could be desirable. In such cases an adaptive system (adjusting to the current conditions of the system, such as temperature) could be necessary.

Here we will describe the adaptive polynomial suppressor of geometrical distortions of CRT displays (or other imaging systems which could have such kind of distortions).

There are two approaches to the problem of geometrical distortion suppression: the bilinear correction and the polynomial correction (see [38]). Since the geometrical distortions are weak, the polynomial approach (see the above equations) seems to be more rational.

Figure 67. An example of the geometrically distorted image.

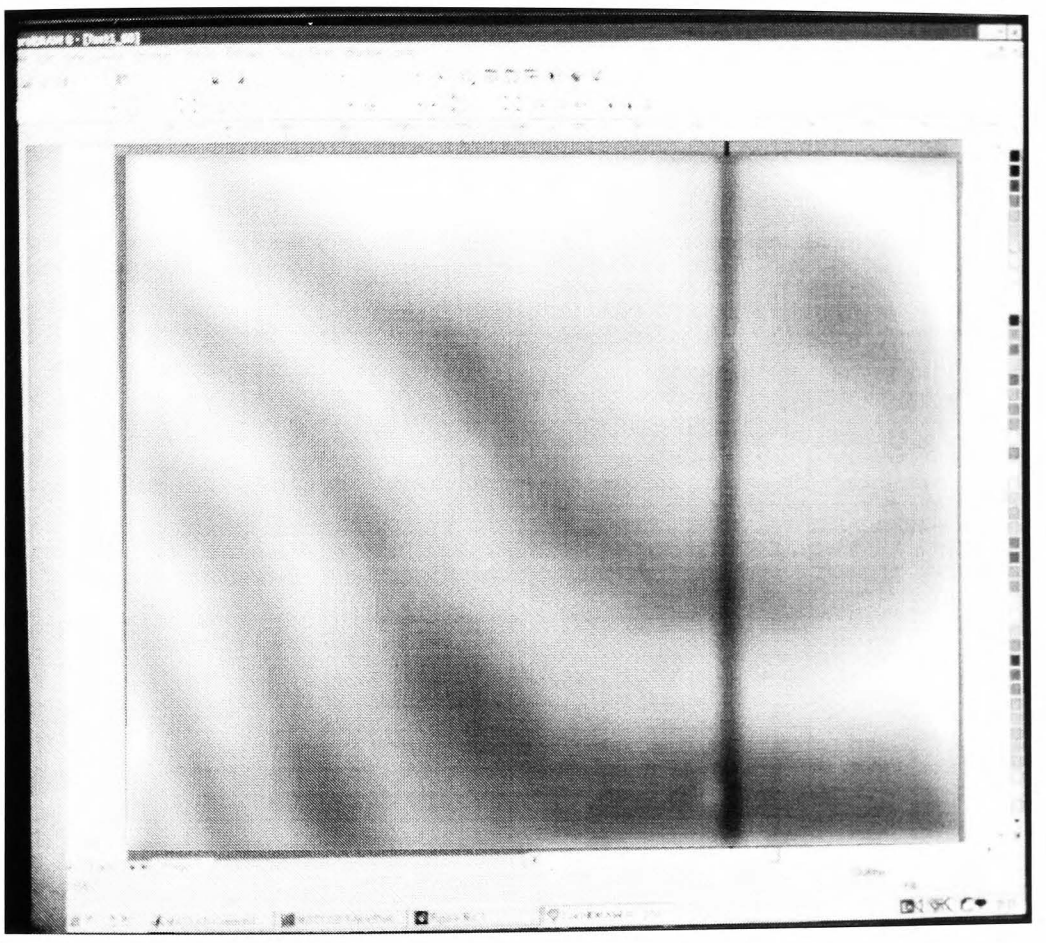


Figure 68. The grating (consisting of vertical strips) could be used for measuring the geometrical distortions of a display. Here the bright and dark areas are related to the match and mismatch of the strips of the grating and the vertical lines pictured by the display.

3.6.1 Description of the system.

The suppressor of geometrical distortions consists of

1) an optical unit, generating the "parameter of quality" δ, which is used by

2) an electronic unit for generation of predistortions of the "swipe" signals (see Figure 69).

3) an electronic system, forming a test image shown by a display during the procedure of adaptation.

The "parameter of quality" is generated by the optical unit including:

1. lens
2. mask with uniformly distributed in space small holes
3. photodiode and integrator

The maximisation of the "parameter of quality" will allow us to reach the goal: the suppression of the geometrical distortion by means of the following procedure.

3.6.2 The function of the system.

The system is shown in Figure 69. It implements the equation (3.37a) describing the polynomial corrected horizontal sweep signal x' (a similar system could be used for the vertical sweep signal y' correction).

The optical system, which gives us the "parameter of quality", implements the following equation: $\delta = \sum_{i=1,j=1}^{N,M} I(i,j)$, where δ is an integrated signal taken from the photodiode, $I(i,j)$ is an intensity of light going through $(i,j)^{th}$ hole of the mask. The mask is placed in position where the image of display is focused, thus the $I(i,j)$ is proportional to an intensity of related point on the display. This intensity is related to the intensity of the central part of the $(i,j)^{th}$ spot on the screen $I_0(i,j) = I_0$ in accordance with the formula: $I(i,j) = I_0 \exp\left(-\frac{(x-x_{ij})^2+(y-y_{ij})^2}{r^2}\right)$. This means that the "training image", displayed on the screen, consists of uniformly spatially distributed white spots with the Gaussian distribution of the intensity on the black background. The parameter r is the size of spots, which is defined from the condition:

$$r = 4\frac{\sum_{i=1,j=1}^{N,M} I(i,j)}{N \cdot M \cdot I_0}. \tag{38}$$

So the dynamics of the display correction is the following. In the initial state, when the weights of the system are zero and the geometrical distortions are not sup-

pressed, the test picture on the screen consists of white spots of big size. After the start of the process of correction, the system alternately adjusting its degrees of freedom, which corresponds to the altering of polynomial coefficients.

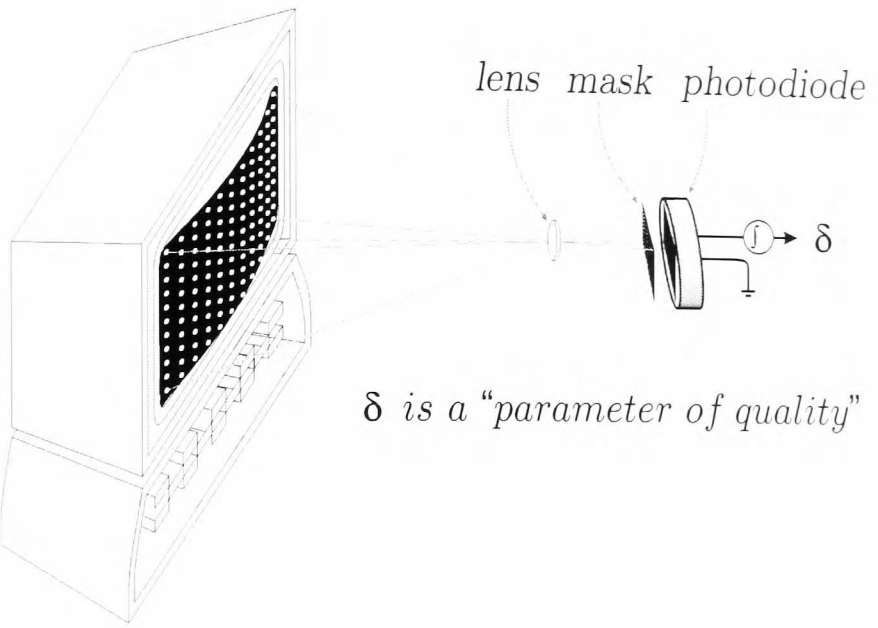
During the process of the system's adjustment, the parameter of quality of the system is increasing, which means that the more and more bright parts of the spots are projected on the holes of the mask. This means that the geometrical distortions of the display are decreasing. To maximise the signal-to-noise ratio, during the adaptation of the system the sizes of spots decrease in accordance with equation 3.38.

The dynamics of the display adjustment is shown in Figure 70 (note that in this figure the "test image" is negative; in a real system it should be white spots and black background). In the initial stage of adjustment the geometrical distortions are substantial. We can see that during the adaptation, along with decreasing of the geometrical distortions, the sizes of spots are also decreasing.

It should be noted that the system shown in Figure 69 is implementing the equation 3.37a indirectly. Instead, it uses the Legendre polynomial 2D function basis for the geometrical distortion approximation and suppression. The orthogonality of the Legendre polynomial 2D function basis gives a huge gain in speed of adaptation (see Figure 71). The minimal time necessary for the Legendre polynomial-based system adaptation is about a half a minute, whereas in the case of the power-type functions, the adaptation time could be estimated as at least 15 min. An another advantage of the Legendre-based system is its higher noise tolerance and stability of convergence. The dynamics of the process of the geometrical distortion suppression is illustrated also in Figure 72.

3.6.3 Implementation.

The described system is rather complex (it contains a lot of multipliers). Both hybrid analogue/digital and analogue implementations of the nonlinear sweep-signal correction system (shown in Figure 69) are possible. In the case of a hybrid system however the bilinear correction could be more suitable (since the bilinear transform could be easyer implemented in analogue domain in comparison with the polynomial transform, whereas the system defining the parameters of the bilinear transform seems to be difficult to implement in analogue domain).



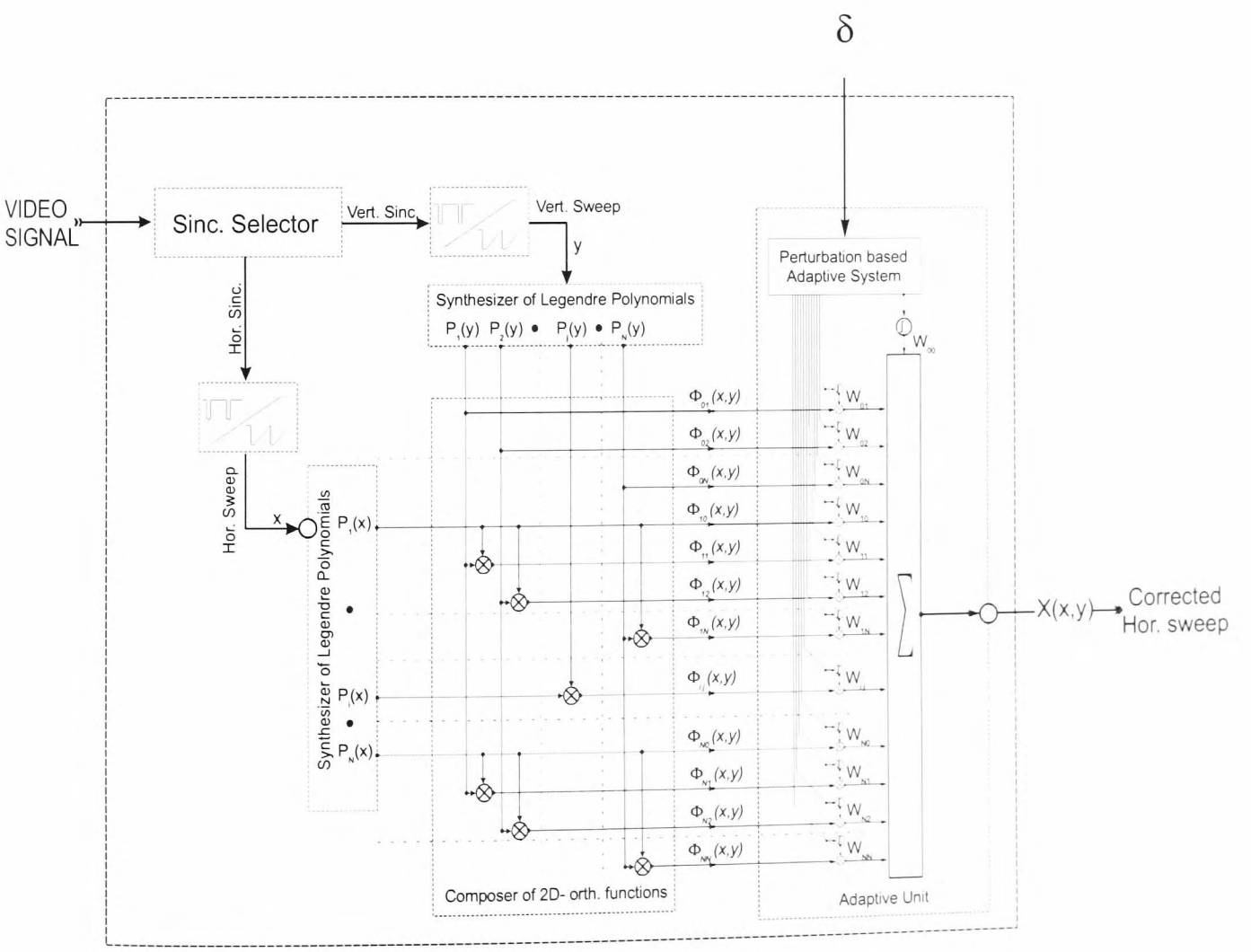
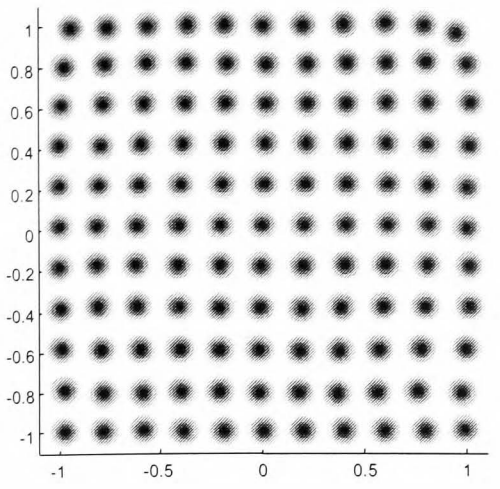
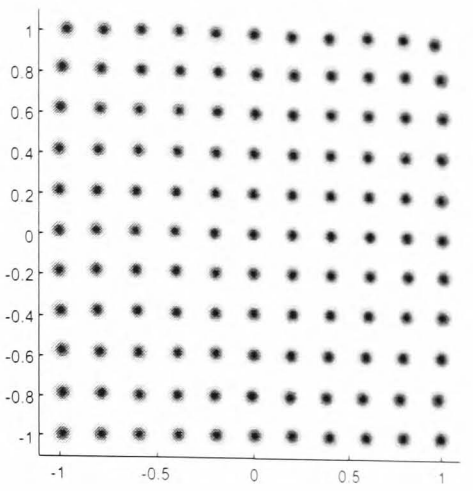


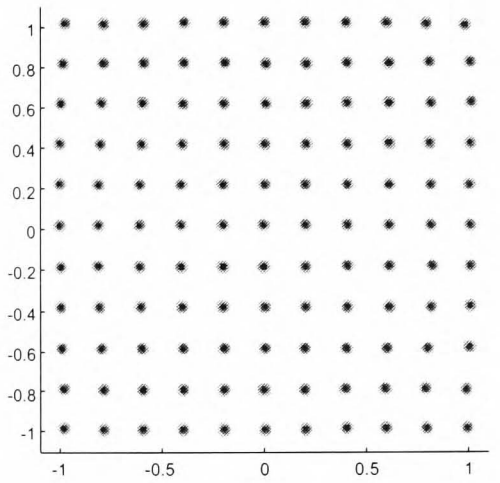
Figure 69. The suppressor of geometrical distortions consists of
1) an optical unit, generating the "parameter of quality" δ (the upper image) which is used by
2) an electronic unit generating the predistortions of the "swipe" signals (the lower image)



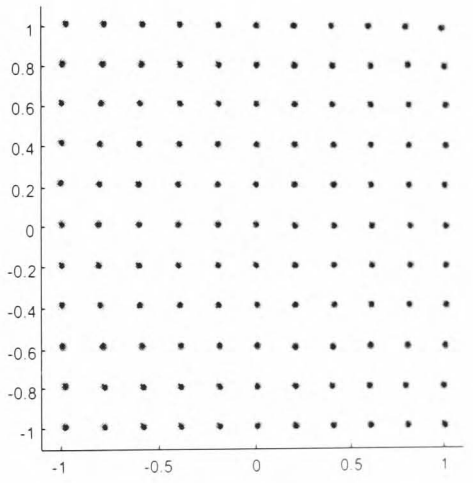
(a) k=5



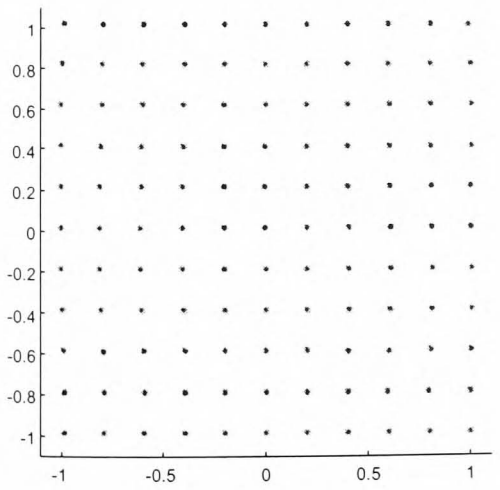
(b) k=10



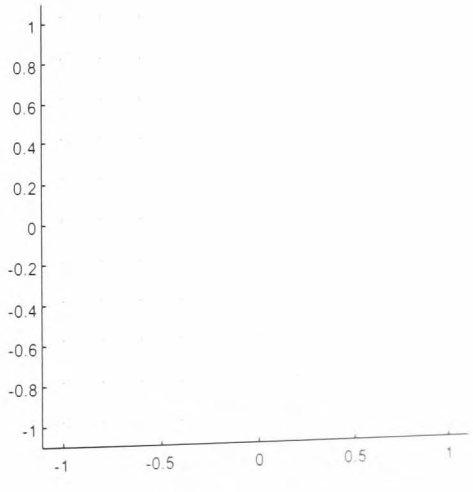
(c) k=15



(d) k=20

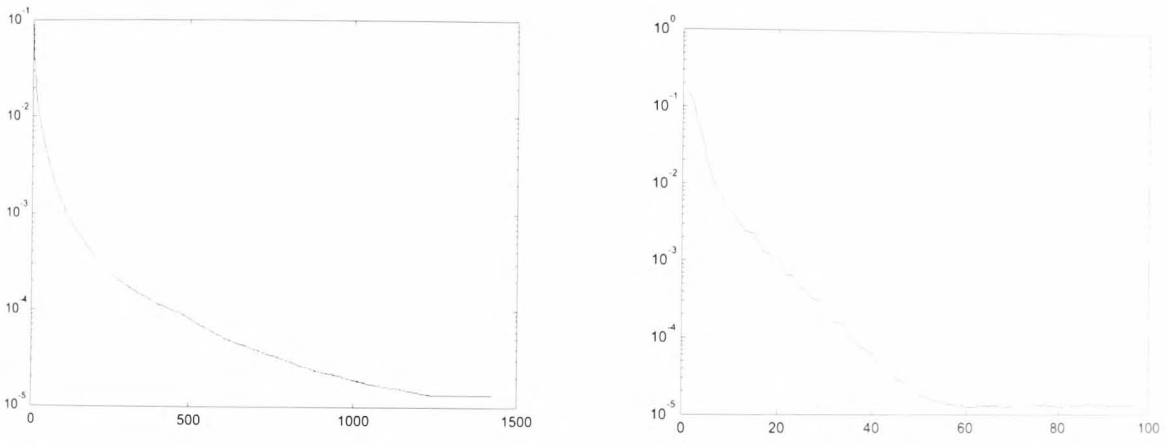


(e) k=30



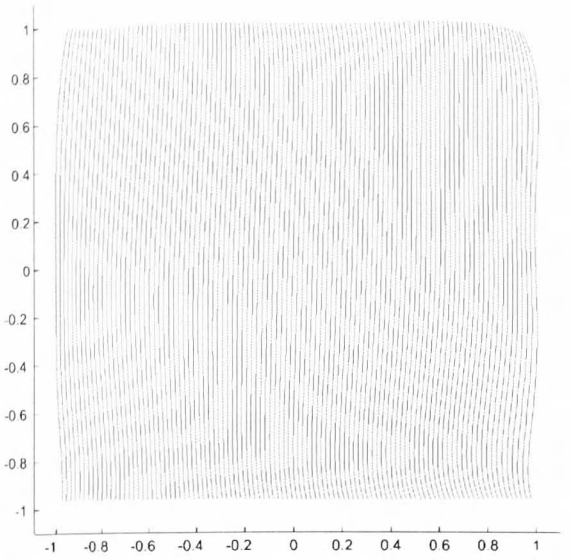
(f) k=100

Figure 70. The dynamics of display adjustment. In the initial stage of adjustment the geometrical distortions are substantial. During the adaptation, along with decrease of the geometrical distortions, the sizes of the spots are also decreasing.

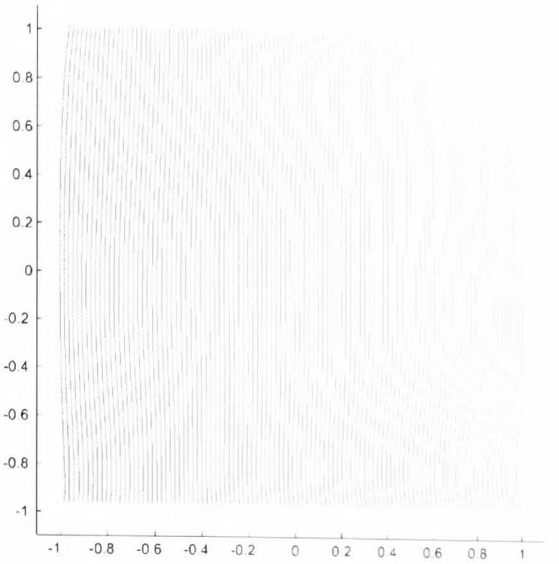


(a) (b)

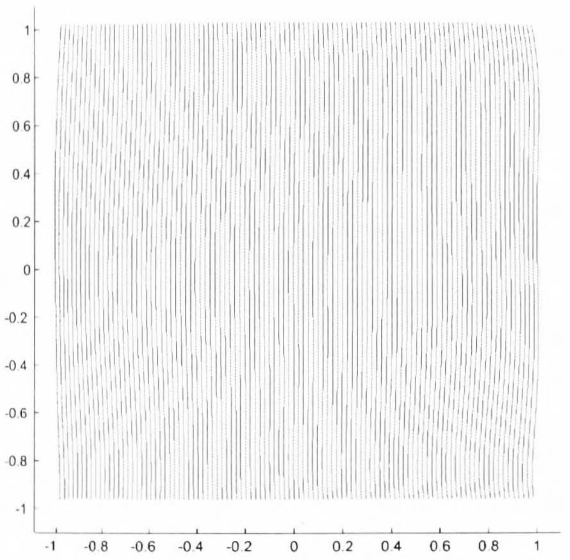
Figure 71. Comparison of the convergence curves for the cases of a) power type and b) Legendre polynomial bases of functions used for geometrical distortion suppression.



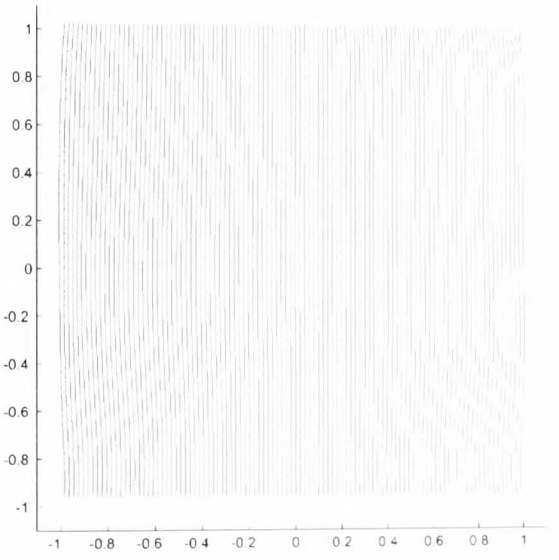
k=5



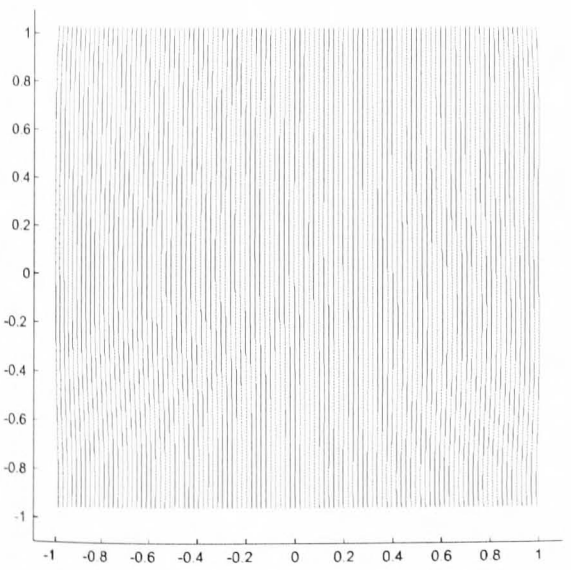
k=10



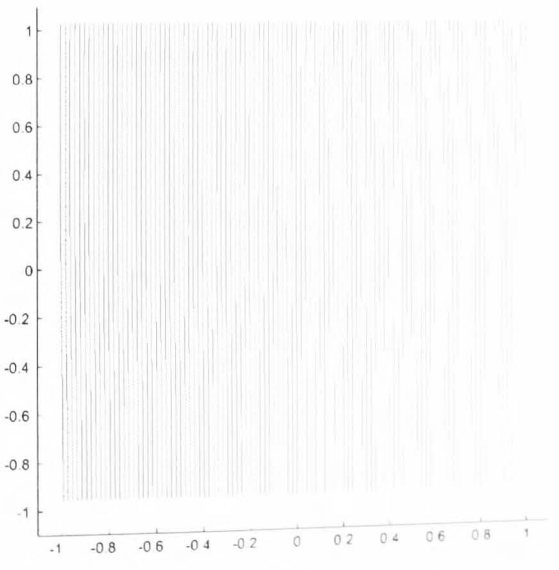
k=15



k=20



k=30



k=100

Figure 72. Dynamics of the process of geometrical distortion suppression.

The Legendre polynomials for analogue implementation of the polynomial correction system could be synthesised in the analogue domain by means of the system described in section 3.2.

On the other hand the basic functions could be power type functions, but they should be perturbed in such way that the output signal would have a Legengre-polynomial increment. This implies that during a perturbation step not one but several coefficients of the correcting polynomial W_i are to be changed.

The results of the adaptation could be stored into a digital memory.

The perturbation unit should be implemented in analogue to avoid the noise of digitising and to maximise the precision of the system (an example of a perturbation-based analogue multiparameter adaptation system is shown in Figure 50).

The system seems quite complex, but it has a very important advantage: the adaptivity. Thus if the parameters of the display have degraded over time for some reason (temperature or long term degradation), the correction system will be able to suppress the distortions. We believe that in case of analogue implementation of the system it could be a special, not too expensive microchip.

The technique described above could be used also for a display calibration (for luminescence spatial non-uniformity suppression).

3.7 Analogue parallel-learning MLP neural network

It is a well known fact that natural neural systems are using analogue (not digital) mechanisms/processes (nonlinear physical, chemical, electrochemical, biological phenomena) for information processing (see e.g. [9]). This gives an enormous gain in compactness and in power consumption of the natural neural networks in comparison with artificial digital systems of information processing. This fact is one of the main motivations for research toward developing neuromorphic systems.

Normally neural networks (both natural and artificial) consist of a feedforward part (the inputs of neural networks (e.g. signals from receptors of natural NN) are applied to inputs of feedforward networks) and feedback part, which has the function of modification of the feedforward network (normally the instantaneous error is on the inputs of the feedback network). In Figure 73 the scheme of the Multilayer Perceptron ANN is shown. The arrows directed from left to right are symbolising the feedforward network, whereas the arrows directed from right to left are symbolising the learning system.

In Figure 74 the schemes of analogue neurons (from input, hidden and output layers) constituting the backpropagation learning ANN (shown in Figure 73) are represented. The multipliers **A**, **B** and **D**, the upper summator, integrators and nonlinear

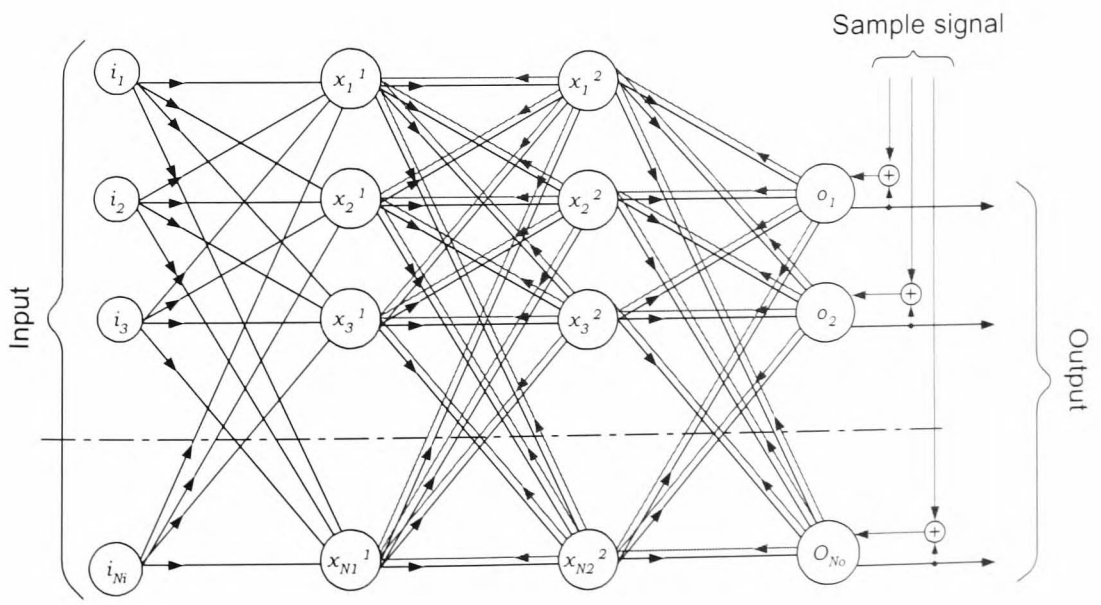


Figure 73. Scheme of the Multilayer Perceptron ANN with backpropagation learning.

elements $f\prime(s_j)$ are related to the learning system, whereas the analogue multipliers **X**, lower summators and nonlinear elements $f(s_j)$ are related to the feedforward network.

In most cases the feedforward part of ANN could be imprecise, whereas the backward part of the ANN must be extremely precise (generally speaking the necessary precision of analogue ANN depends on the task the ANN must solve as well as on a particular neural architecture. Typical necessary precision is about 10^{-6}, see e.g. [51], see also the subsection 3.7.1). The offsets of different elements of neurons are symbolised in Figure 74 as signs $\oplus$.

It is interesting that noise is undesirable in the feedforward network (after learning) whereas the noise of the feedback network could be even useful during the learning of ANN. It should be noted here that the question of tolerance of the feedforward network to imprecision of its elements is not trivial: in subsection 3.7.2 the influence of different offsets of elements of a feedforward network is analysed (the learning system is assumed to be ideal).

It is known that any analogue element is imprecise (because of mismatches of characteristics of semiconductor elements, e.g. it is impossible to make two absolutely equivalent transistors). The question is: **how it is possible to create a high precision system on the basis of imprecise elements?** Such systems could be created, but

1. they must contain a special subsystem for correction of imprecise elements (let's call it *self-correction system*);

2. a special phase of the ANN function, the *self-correction phase,* is necessary (during this phase the imprecisions of the learning system must be suppressed).

If we take a look at natural neural networks, they obviously have such a kind of self-correction system and procedure. The procedure of self-correction of natural neural networks might be taking place **during sleep**.

As for artificial systems, until now most ANN didn't contain on-chip (or on-board) analogue **parallel** learning systems.

We should mention here, however, the purely analogue MLP ANNs with perturbation–based learning systems. Such systems are based on either serial weight perturbation (see [52] and [53]) or on the combined outputs and weights perturbation technique

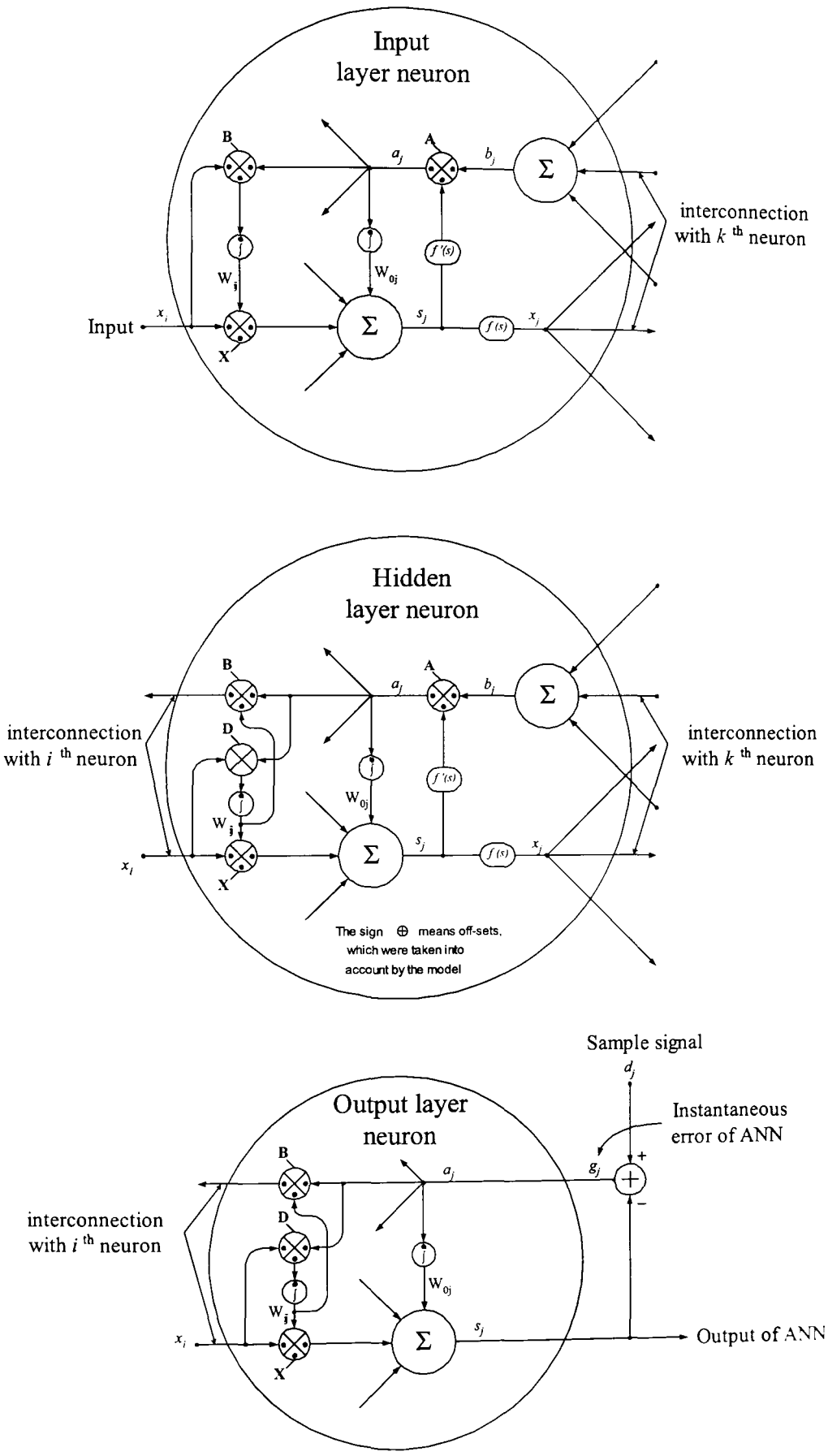


Figure 74. Schemes of analogue neurons (from input, hidden and output layers) constituting the backpropagation learning ANN.

(described in [51]). Such perturbation-based systems are very tolerant of analogue nonidealities. On the other hand they are much slower at learning (e.g. the serial weight perturbation-based system is N times (N is the number of interconnections, i.e. synapses) slower than the fully parallel backpropagation learning system). In the case of complex neural networks (with e.g. $N = 10^4$) such a drawback of perturbation-based analogue ANNs could make them useless.

That's why the problem of purely analogue parallel-learning ANN is very interesting and important.

The problem of creation of high precision ANN based on imprecise elements is not trivial. The trivial self-correction system, based on *self-correction of separate elements*, such as differential amplifiers, integrators, analogue multipliers, etc. (let's call such system a *"local" self-correction system*), doesn't give the maximal possible precision. Indeed it could be many times worse than in case of other self-correction systems. Besides, in some cases (when the interconnections bring in additional off-sets) it can not be used at all. Furthermore *"global"* self-correction systems (including several elements and interconnections into the off-sets zeroing circuits) are much simpler than "local" ones. A more detailed analysis of "local" self-correction systems is in subsection 3.7.3.

In this work two kinds of "global" self-correction systems are presented:

1. The self-correction system of the first kind (described in the subsection 3.7.4) allows the proper learning of ANN after the self-correction procedure as long as parameters of system (like offsets) or correcting constants (supposed to be held by capacitors) are unchanged. The weights however are allowed to vary. This means that the learning of ANN doesn't affect the precision characteristics of learning system.
2. The self-correction system of the second kind (described in subsection 3.7.5) is considerably simpler than the system of the first-kind. On the other hand, the learning system corrected by this self-correction system *is vulnerable to changes of parameters of ANN*. That is, the learning system, adjusted by the self-correction system of the second kind, is highly precise *only for current particular weights of ANN*. **Nevertheless such a system could be successfully used** if the process of learning of ANN with this system of self-correction is alternating with the self-correction procedure. In practice, this could be e.g. one or several cycles of learning alternating with one or several cycles of self-correction.

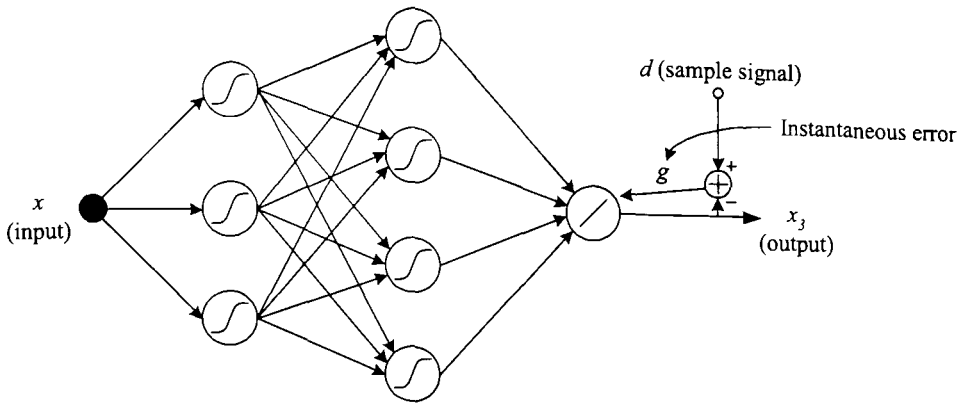


Figure 75. Neural architecture used in the simulations

All the above-mentioned neural architectures were modeled on the computer. It should be noted here that **nonlinear distortions of elements of the learning system, as well as the signal propagation delay, were not taken into account by the computer models.**

The ANN with the self-correction system of the second kind was implemented in a proof-of-concept breadboard.

General Neural Architecture and the Neural Task. Let us describe the neural architecture and the neural task which were used in the simulations of different self-correction schemes. The neural network has one input, three neurons in the first (input) layer, four neurons in the second (hidden) layer and one output neuron. All neurons have hyperbolic tangent transfer function, except the output neuron, which has a linear transfer function (see Figure 75).

The internal architectures of neurons are shown in Figure 74. For the *case of ideal ANN* we will have:

$$\mathbf{x}_1 = f\left(\mathbf{W}_1 x + \mathbf{W}_1^0\right);$$

$$\mathbf{x}_2 = f\left(\mathbf{W}_2 \mathbf{x}_1 + \mathbf{W}_2^0\right);$$

$$x_3 = \mathbf{W}_3 \mathbf{x}_2 + \mathbf{W}_3^0;$$

(here the bold symbols mean vectors or matrixes). The instantaneous error signal $g \equiv d - x_3$, where d is the sample signal. The backpropagation learning system function in

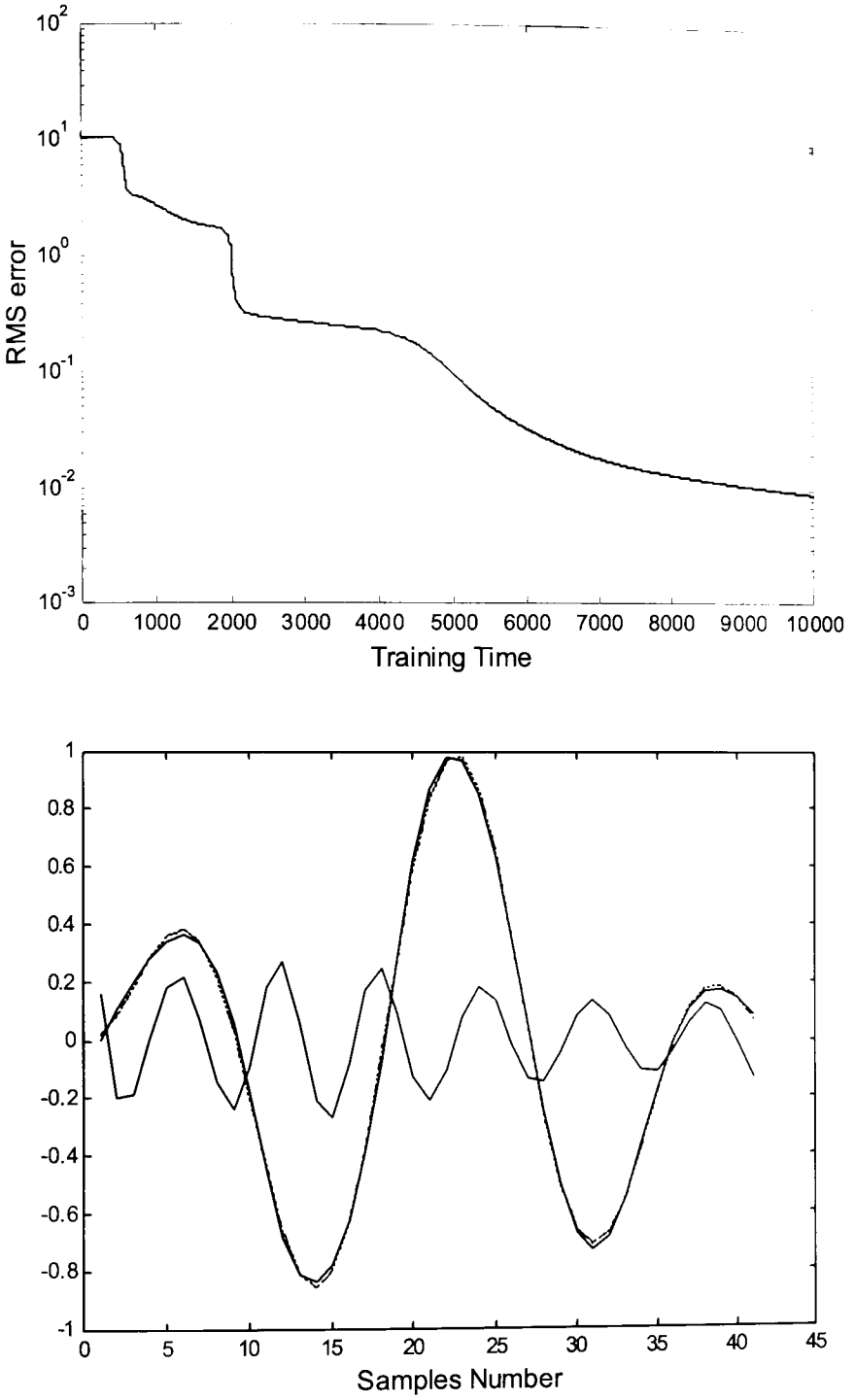


Figure 76. Upper plot: error curve for the ideal ANN. Lower plot: sample signal (dashed line), the output signal (solid line) and the instantaneous error signal multiplied by 10.

the *case of ideal ANN* is described by the following equations (see Figure 74):

$$a_3 = g = d - x_3;$$

$$\mathbf{a}_2 = f'(\mathbf{s}_2)(\mathbf{W}_3^T a_3);$$

$$\mathbf{a}_1 = f'(\mathbf{s}_1)(\mathbf{W}_2^T \mathbf{a}_2);$$

$$\Delta \mathbf{W}_3 = \alpha \int_t^{t+\Delta t} a_3 \mathbf{x}_2^T dt; \qquad \mathbf{W}_3 = \mathbf{W}_3 + \Delta \mathbf{W}_3;$$

$$\Delta \mathbf{W}_2 = \alpha \int_t^{t+\Delta t} \mathbf{a}_2 \mathbf{x}_1^T dt; \qquad \mathbf{W}_2 = \mathbf{W}_2 + \Delta \mathbf{W}_2;$$

$$\Delta \mathbf{W}_1 = \alpha \int_t^{t+\Delta t} \mathbf{a}_1 x dt; \qquad \mathbf{W}_1 = \mathbf{W}_1 + \Delta \mathbf{W}_1;$$

$$\Delta W_3^0 = \alpha \int_t^{t+\Delta t} a_3 dt; \qquad W_3^0 = W_3^0 + \Delta W_3^0;$$

$$\Delta \mathbf{W}_2^0 = \alpha \int_t^{t+\Delta t} \mathbf{a}_2 dt; \qquad \mathbf{W}_2^0 = \mathbf{W}_2^0 + \Delta \mathbf{W}_2^0;$$

$$\Delta \mathbf{W}_1^0 = \alpha \int_t^{t+\Delta t} \mathbf{a}_1 dt; \qquad \mathbf{W}_1^0 = \mathbf{W}_1^0 + \Delta \mathbf{W}_1^0,$$

where α is a parameter of speed of learning. The task for the ANN is the approximation of a smooth function:

$$d(x) = \sin(7x + 1)\cos(1.5x)$$

in the range $-1 < x < 1$. Here we assume that x is a periodical signal with period Δt and with linear dependence as a function of time t within each period.

For such a case the convergence curve is shown in the upper plot of Figure 76. The result of learning of the ANN is shown in the lower plot of Figure 76.

The self-correction systems, which will be described in the following subsections, are directly related to possible continuous analogue implementations of ANN. Nevertheless, the obtained results could be generalised to the cases of pulse-encoding based analogue ANN.

3.7.1 Purely Analogue ANN without Self-correction System

It is well-known that the learning systems of supervised neural networks (like Multilayer Perceptron ANN with backpropagation learning) must be very precise (see e.g. [51]).

If the learning system is not precise, normally as a result of learning the neural network doesn't converge to the global minimum of error. The behavior of ANN with such imprecise learning systems is normally the following: at the beginning of learning

the error is decreasing in approximately the same way as it is in the case with an ideal learning system, but after a while at some stage the error starts to grow. In Figure 77 a typical example of the learning curve is shown. It is possible to see that at some stage the system could even become unstable (see the burst on the learning curve).

The result of learning of such systems normally is unacceptable (see the lower graphs in Figure 77). However the results could be in some cases acceptable if the learning was stopped at the point of minimal error (see e.g. [51]).

Let us describe the model, which was used for simulation of such systems. Let us first represent the *ideal* neuron in terms of its particular components:

$$x_j = f\left(\sum_i W_{ij}x_i + W_j^0\right);$$

$$a_j = f'(s_j)\sum_k W_{jk}a_k; \tag{39}$$

$$\Delta W_{ij} = \alpha\int_t^{t+\Delta t} a_j x_i dt; \qquad W_{ij} = W_{ij} + \Delta W_{ij}; \tag{40}$$

$$\Delta W_j^0 = \alpha\int_t^{t+\Delta t} a_j dt; \qquad W_j^0 = W_j^0 + \Delta W_j^0; \tag{41}$$

Let us consider now the *nonideal* neuron. The nonideality of neurons we will describe by adding offsets to analogue multipliers **A** (A_x, A_y, A_z), **B** (B_x, B_y, B_z), **D** (D_x, D_y, D_z) and **X** (X_x, X_y, X_z) (the offsets of integrators were supposed to be included in the related offsets of multipliers connected to these integrators), see Fig 78. **The nonlinear distortions of elements as well as time delays were not included into the model.**

$$x_j = f\left(\sum_i \left(\left(W_{ij} + X_y^{(ij)}\right)\left(x_i + X_x^{(ij)}\right) + X_z^{(ij)}\right) + W_j^0\right); \tag{42}$$

$$a_j = \left(f'(s_j) + A_y^{(j)}\right)\left(\sum_k \left(\left(W_{jk} + B_y^{(jk)}\right)\left(a_k + B_x^{(jk)}\right) + B_z^{(jk)}\right) + A_x^{(j)}\right) + A_z^{(j)}; \tag{43}$$

$$\Delta W_{ij} = \alpha\int_t^{t+\Delta t}\left(\left(a_j + D_y^{(ij)}\right)\left(x_i + D_x^{(ij)}\right) + D_z^{(ij)}\right)dt; \quad W_{ij} = W_{ij} + \Delta W_{ij}; \tag{44}$$

$$\Delta W_j^0 = \alpha\int_t^{t+\Delta t}\left(a_j + D_a^{(j)}\right)dt; \qquad W_j^0 = W_j^0 + \Delta W_j^0.$$

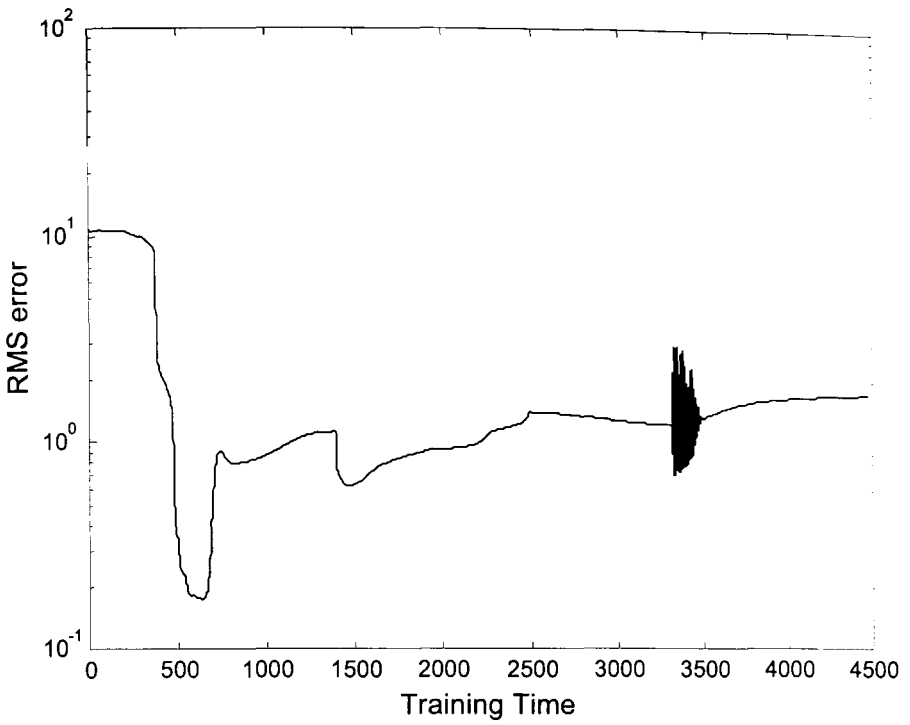


Figure 77. Upper plot: error curve for the nonideal ANN (offsets are about 10^{-3}). Lower plot: sample signal (dashed line) and the output signal (solid line).

The results of simulation of the system described by the above equations with randomly distributed offsets (average offset 10^{-3}) are shown in Figure 77.

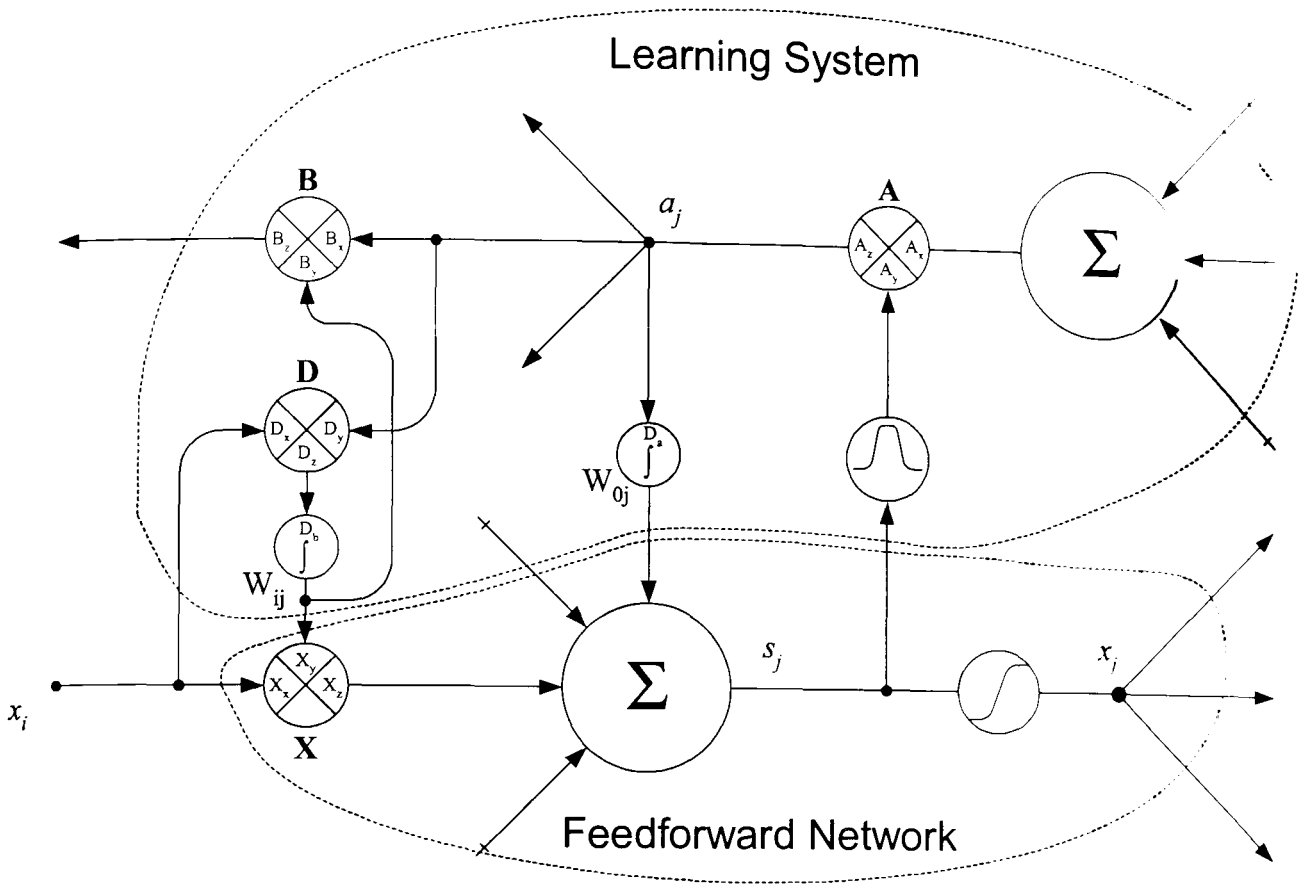


Figure 78. Backpropagation neuron with parts marked related to learning system and to feedforward network.

3.7.2 ANN with imperfect (analogue) feedforward network and perfect (digital) learning system

It is widely accepted that, unlike the learning system, the feedforward network of ANN could be imprecise (i.e. could contain offsets and nonlinear distortions).

Such systems (with ideal learning system and nonideal feedforward network) were modeled and the above statement was confirmed. During the simulation (based on the above-mentioned neural architecture 1x3x4x1 and neural task) the offsets of feed-forward network (X_x, X_y, X_z) (see Figure 78) were larger than 10^{-2} and this didn't affect either the stability and speed of convergency, or the final error approximation (the offsets to analogue multipliers **A** (A_x, A_y, A_z), **B** (B_x, B_y, B_z) and **D** (D_x, D_y, D_z) related to the learning system were assumed to be equal to zero).

The question of ANN's tolerance of feedforward network nonidealities is of great importance since it determines whether the creation of analogue ANN, with off-chip learning as well as with on-chip digital learning system, is possible or not . In fact this question is not trivial.

If we considere the nonideal feedforward network (see Eq. 3.42):

$$\begin{aligned} x_j &= f\left(\sum_i \left(\left(W_{ij} + X_y^{(ij)}\right)\left(x_i + X_x^{(ij)}\right) + X_z^{(ij)}\right) + W_j^0\right) \\ &= f\left(\sum_i \left(W_{ij}x_i + W_{ij}X_x^{(ij)} + X_y^{(ij)}x_i + X_y^{(ij)}X_x^{(ij)} + X_z^{(ij)}\right) + W_j^0\right); \end{aligned}$$

we could see that the offsets $X_x^{(ij)}$ and $X_z^{(ij)}$ are not affecting the characteristics of ANN since their influence will be easily suppressed by the bias weights W_j^0. The questions arise in connection with the offsets $X_y^{(ij)}$, since they are responsible for the term $X_y^{(ij)}x_i$.

Mathematical analysis shows that it is very difficult to describe the influence of offsets $X_y^{(ij)}$ to the behavior of ANN. It is relatively easy to prove that the offsets $X_y^{(ij)}$ are not affecting the neurons of the input ANN layer. As for the hidden and output layer, their vulnerability to such offsets needs to be described (the author is unaware of any such works in the literature).

The speculative analysis of the influence of such offsets shows that if $\sum_i X_y^{(ij)}x_i \sim 1$, it could degrade the properties of ANN. On the other hand it is easy to prove that the *global minimum* of the mean square error of ANN $\mathcal{E}_0$ (where the mean square error $\mathcal{E} \equiv \int_{\Delta t}\left(\sum_k g_k^2(t)\right)dt$) doesn't depend on the offsets $X_y^{(ij)}$ in a linear approximation:

$$\mathcal{E}_0|_{X_y^{(ij)}=\delta} \underset{\delta\to 0}{=} \mathcal{E}_0|_{X_y^{(ij)}=0}.$$

This means that the global minimum of error $\mathcal{E}_0(X_y^{(ij)})$ (in case of ANN with *nonideal* feedforward network) as a function of parameters $X_y^{(ij)}$ has a global minimum in parameters $X_y^{(ij)}$ space, and this minimum is at $X_y^{(ij)} = 0$.

3.7.3 ANN with Trivial Self-correction System ("local" self-correction)

The trivial self-correction system implies the *independent* and *individual* self-correction of all active elements of ANN (analogue multipliers and summators) that ought to be precise. Let's call such self-correction systems "local".

The basic element of self-correction systems is a Voltage-to-Current convertor. In Figure 79 inverting (a) and non-inverting (b) converters of voltage-to-current are shown.

In the self-correction mode the inputs of the converters are switched to the zero level, whereas the outputs are switched into correcting feedback. In the steady state, the current doesn't flow and the converter is adjusted. If we will put a capacitor (with

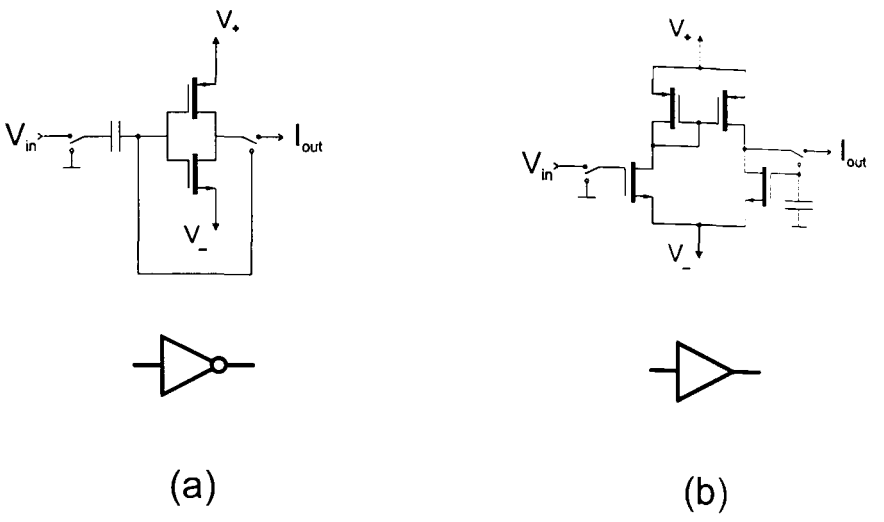


Figure 79. Inverting (a) and non-inverting (b) converters of voltage-to-current.

other contact grounded) on the output of such converters, we will get a self-correcting integrator.

The procedure of correction of analogue multipliers is well-known:

- first stage: both x and y inputs are connected to ground. By means of the correcting input z_0 of the multiplier one can achieve the zero output z of the multiplier. The z_0 established must be fixed;
- second stage: x input is connected to the ground, whereas the y input is connected to a reference voltage V_+ (let's say 1 V). By means of the correcting input x_0 of the multiplier one can achieve the zero output z of the multiplier. The established x_0 must be fixed;
- third stage: y input is connected to the ground, whereas the x input is connected to a reference voltage V_+. By means of the correcting input y_0 of the multiplier one can achieve the zero output z of the multiplier. The established y_0 must be fixed.

If necessary, such a procedure could be repeated several times, thus the offsets of the multiplier could be strongly enough suppressed (see Figure 80b). The schematic of the self-correction multiplier is shown in Figure 80a. We can see that any active element of ANN could be adjusted separately. The question is whether it is maximal possible precision or not. It can be easily shown that it is not.

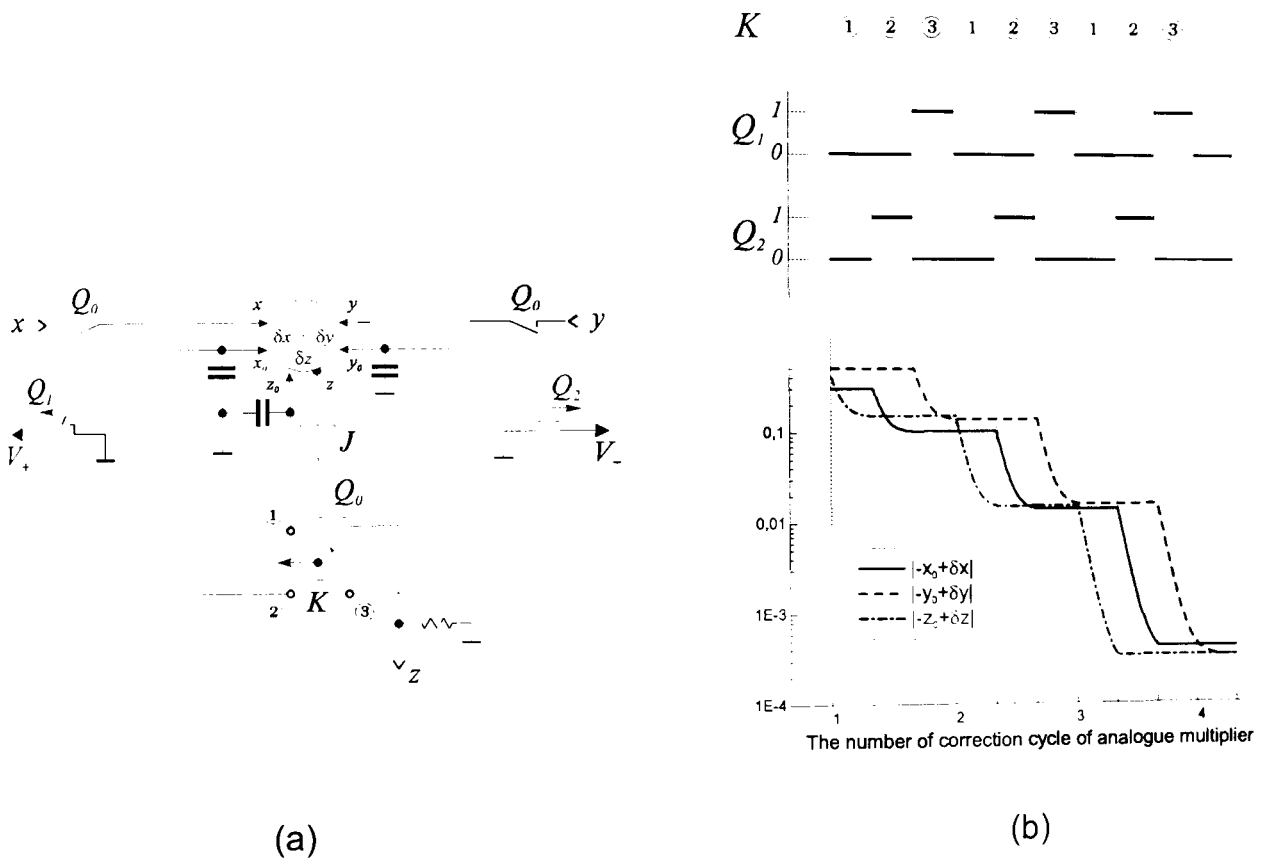


Figure 80. (a) electrical scheme and (b) diagram explaining the procedure of correction of analogue multiplier.

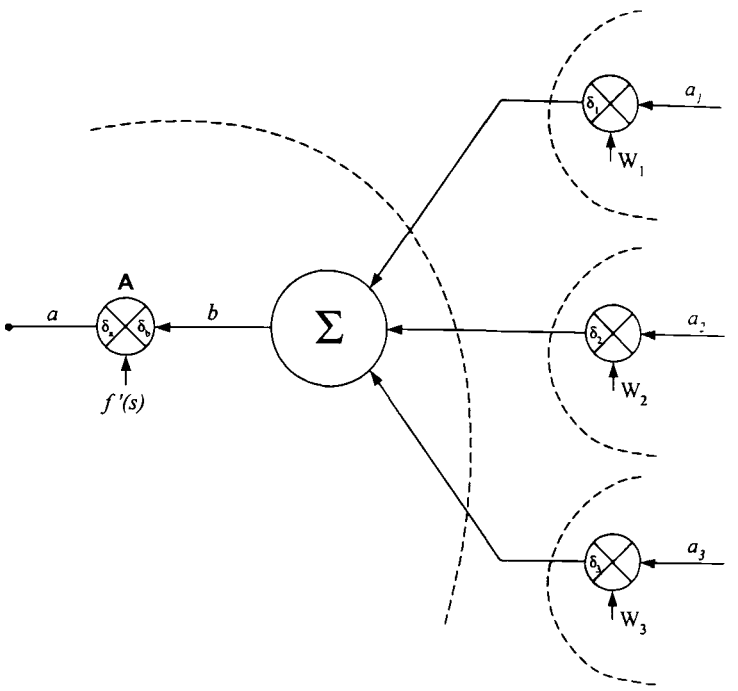


Figure 81. The signal b on the output of summator of this part of the learning system is accummulating the remanent errors of the preceding analogue multipliers δ_k.

First of all, the above described systems, even after the self-correction procedure, are not ideal. Still there are some remaining errors caused e.g. by the capacitance of switches. Figure 81 illustrates such a drawback of the "local" self-correction system as a not maximal possible precision of the learning system. The offset δa of the signal a (which is $\delta a \equiv a - a_0$, where a_0 is a signal a in case of ideal learning system) is accumulating several nonidealities of multipliers shown in the right hand side of Figure 81, in particular by their z-offsets: δ_1, δ_2, δ_3, in accordance with the formula:

$$\delta a = f'(s)(\delta_b + \delta b) + \delta_a + ... = f'(s)\left(\delta_b + \sum_{k=1}^{3} \delta_k\right) + \delta_a + ... \quad .$$

Since the function $f'(s)$ could be ~ 1, and taking into account that the remanent errors of multiplier **A** and multipliers of the preceding neurons are the same, we can see that, in case of a large number of interconnections, the contribution of offsets from the preceding neurons could be much bigger than the remanent errors δ_a and δ_b of the multiplier **A**. For example if the number of interconnections k is 100, in case of randomly distributed offsets, the contribution from the preceding neurons could be larger by the order of magnitude than the remanent error of multiplier **A** (see Figure 81).

Another drawback of the "local" self-correction system is that it is not tolerant of the off-sets contributed by interconnections. In case of continuous analogue implemen-

tation of the learning system such offsets could be small (e.g. if information is in the charge current), whereas in the case of frequency-encoded signals, the interconnections could contribute substantial offsets. In such cases the "local" self-correction system is useless.

The final argument against the "local" self-correction system is its complexity: there is no point to adjust independently the offsets δ_1, δ_2, δ_3 in the above example, since they are accumulating on the output of the summator and could be in principle suppressed by the proper adjusting of the correcting signal A_x of the multiplier **A**.

Since such procedures include several elements and interconnections into the correcting circuit, let's call such systems *"global" self-correction systems*.

3.7.4 ANN with Self-correction System of the First Type

In this and the next subsections the "global" self-correction systems will be presented. The suggested systems are tolerant of offsets from analogue multipliers and integrators, as well as from those contributed by interconnections.

There are two possible approaches to the problem of creation of parallel learning analogue ANN.

The first approach implies that after the self-correction procedure the learning system becomes precise and is capable of learning as long as the correcting parameters of the system are steady. Thus, the Self-correction System of the First Type is based on architectures, in which the parameters (correcting coefficients) do not depend on the weights of the ANN, hence are not affected in the result of learning of the ANN (when the weights are changing).

The second approach, in contrast, is based on a very simple analogue parallel-learning system, in which the parameters (correcting coefficients) **do depend** on the weights of the ANN. The suggested system, nevertheless, provides stable learning, employing the fact that the learning procedure of ANN normally is very slow (hence the weights are changing very slightly after a learning cycle) , thus the procedure of learning could be alternating with a self-correction procedure. For example, it could be one or several cycles of learning are alternating with one or several cycles of self-correction. The idea is that after several cycles of learning (until the self-correction procedure occurs) the learning system still is precise enough to provide a decrease of the error. Such

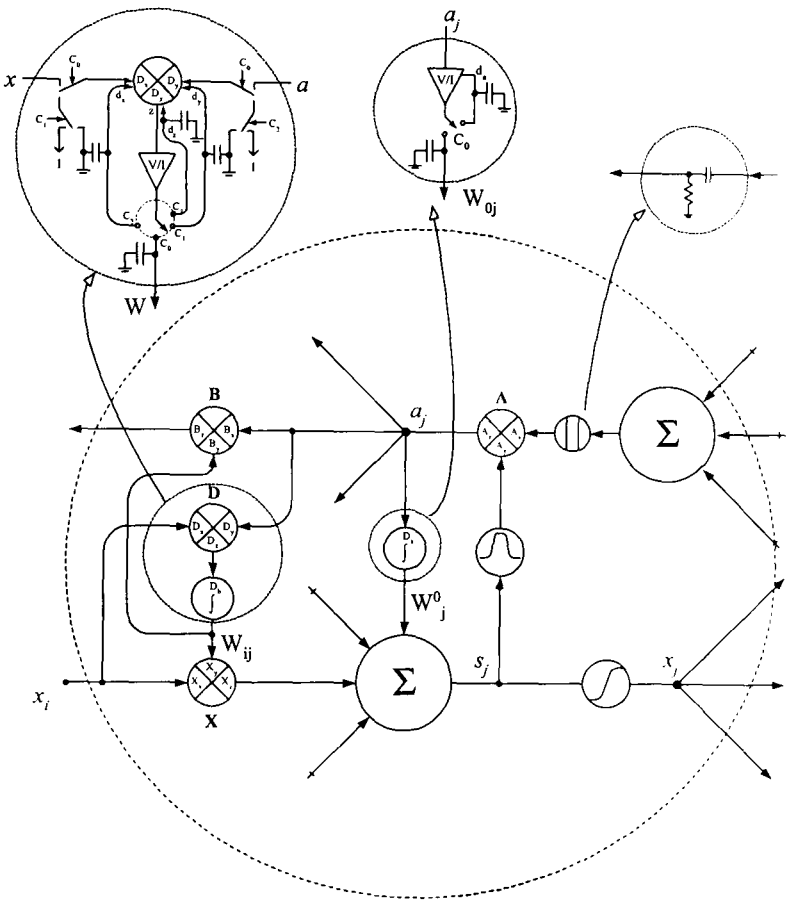


Figure 82. Schematic of ANN with Self-correction System of the First Type, First Kind

an assumption allows us to suggest a very simple and precise analogue parallel-learning system, which is described in the next subsection.

As it was shown in subsection 3.7.2, ANNs are not vulnerable to imprecision of multipliers **X**. Let us consider the influence of other multipliers of the ANN: **A**, **B** and **D**.

We will consider *two kinds* of the self-correction system of the first type:
The first kind: with fully corrected multipliers **D** and **A** and uncorrected multiplier **B** (see Figure 82);
The second kind: with fully corrected multiplier **A** (A_x, A_y, A_z), corrected output's offset of multipliers **D**: $\mathbf{D}_z$ (see Figure 84) and uncorrected multiplier **B**.

***The first type, first kind*: Self-correction system of the first type with fully corrected multipliers D and A, uncorrected multiplier B (see Figure 82).** Let us consider the equation for the convergence of ANN with the imperfect backpropagation learning system described in subsection 3.7.1 (see Eq. 3.43). If we assumed that the multipliers **A** and **D** were adjusted (e.g. by a procedure, described in 3.7.3), we will have:

$$\begin{aligned} a_j &= f'(s_j)\left(\sum_k \left(\left(W_{jk}+B_y^{(jk)}\right)\left(a_k+B_x^{(jk)}\right)+B_z^{(jk)}\right)\right) = \\ &= f'(s_j)\left(\sum_k \left(W_{jk}a_k+B_y^{(jk)}a_k+W_{jk}B_x^{(jk)}+B_y^{(jk)}B_x^{(jk)}+B_z^{(jk)}\right)\right); \end{aligned} \tag{3.45}$$

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} (a_j x_i)\, dt; \qquad W_{ij} = W_{ij} + \Delta W_{ij};$$

To make the learning system equivalent to the ideal one we need to eliminate (suppress) the following terms of Eq. 3.45:

$$W_{jk}B_x^{(jk)} + B_y^{(jk)}B_x^{(jk)} + B_z^{(jk)} \to 0.$$

Such a goal could be achieved by means of the RC circuits, shown in Figure 82. If the time-constant of such RC circuit is less than the time-constant of the integrators, shown in the scheme, the signal on the output of these circuits will free both from constants: $B_y^{(jk)}B_x^{(jk)} + B_z^{(jk)}$ and from the slowly changing term $W_{jk}B_x^{(jk)}$. Since normally the learning process is very slow, such an additional transform of the learning signal will not affect the shape of learning signal $a_j(t)$.

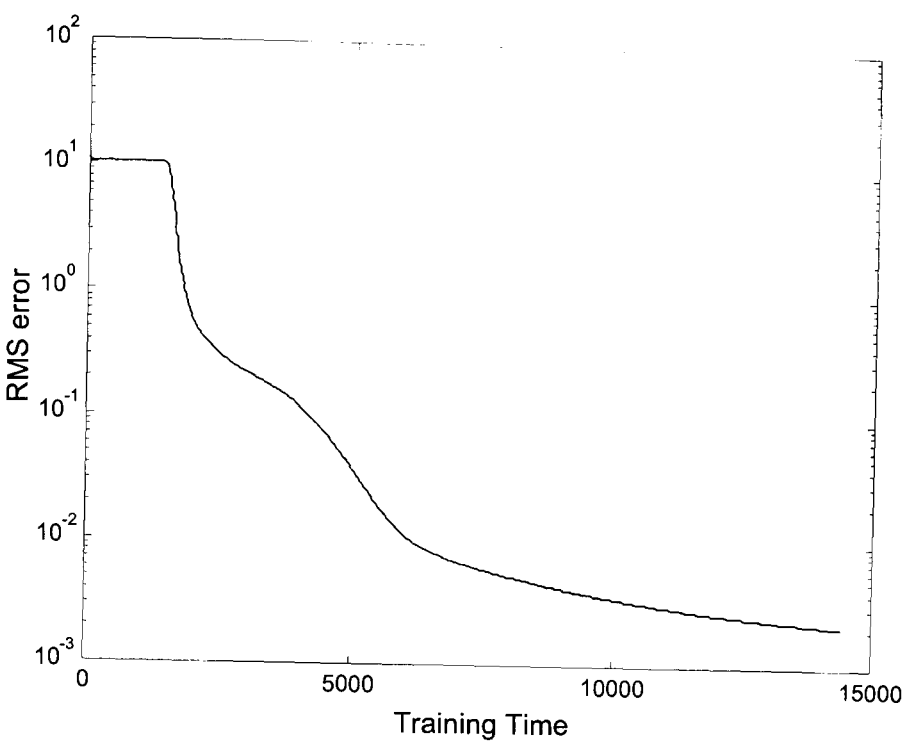


Figure 83. Upper plot: error curve for the nonideal ANN with Self-correction System of the First Type. Lower plot: sample signal (dashed line), the output signal (solid line) and the instantaneous error signal multiplied by 10.

As for the terms $B_y^{(jk)}a_k$ (see Eq. 3.45), it is easy to show, that these terms are not strongly affecting the learning system (since the influence is equivalent to the influence of offsets $X_y^{(ij)}$).

Therefore, the use of such self-correction system allows:

1. to avoid the necessity of correction of multipliers **B**;
2. to avoid the effect of accumulation of remanent errors of **B** multipliers, mentioned in subsection 3.7.3, thus to provide higher precision than the "local" self-correction system could provide.

The results of simulation of analogue ANN with such a self-correction system is shown in Figure 83.

***The first type, second kind*: Self-correction system of the first type with fully corrected multiplier A (A_x, A_y, A_z), corrected output offset of multipliers D: $\mathbf{D}_z$ (see Figure 84) and uncorrected multipliers B and X.** Let us consider the equation for the convergence of ANN with the imperfect backpropagation learning system described in subsection 3.7.1 (see Eq. 3.43). If we assumed that the multipliers **A** were adjusted (e.g. by a procedure, described in 3.7.3), we will have:

$$a_j = f'(s_j)\left(\sum_k \left(W_{jk}a_k + B_y^{(jk)}a_k + W_{jk}B_x^{(jk)} + B_y^{(jk)}B_x^{(jk)} + B_z^{(jk)}\right)\right);$$

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} \left(a_j x_i + a_j D_x^{(ij)} + D_y^{(ij)}x_i + D_y^{(ij)}D_x^{(ij)} + D_z^{(ij)}\right) dt;\ W_{ij} = W_{ij} + \Delta W_{ij};$$

$$\Delta W_j^0 = \alpha \int_t^{t+\Delta t} \left(a_j + D_a^{(j)}\right) dt; \qquad W_j^0 = W_j^0 + \Delta W_j^0.$$

The integrators, shown in Figure 84, should be corrected, in particular:

$$\Delta W_j^0 = \alpha \int_t^{t+\Delta t} \left(a_j + D_a^{(j)} + d_a^{(j)}\right) dt = \alpha \int_t^{t+\Delta t} a_j dt. \tag{46}$$

The use of RC circuits in the learning network, in a similar way as in the previous subsubsection, allows us to suppress the nonidealities of the **B** multipliers and to get for the a_j signal: $a_j = f'(s_j)\sum_k W_{jk}a_k$. A similar trick - an RC-circuit (see Figure 84) allows us to avoid the harmful influence of D_y offsets of multipliers **D:** $D_y^{(ij)}x_i \to 0$. In

this case we have:

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} \left(a_j \widetilde{x}_i + a_j D_x^{(ij)} + D_y^{(ij)} D_x^{(ij)} + D_z^{(ij)} + d_z^{(ij)} \right) dt.$$

The term $\alpha \int_t^{t+\Delta t} a_j D_x^{(ij)} dt \rightarrow 0$ because of the adjustment of the integrator, forming the weights W_j^0 (see Eq.3.46). By means of adjustment of the output offset $d_z^{(ij)}$ of **D** multipliers $D_z^{(ij)}$ (or, which is the same, the input offsets of integrators) one can achieve the zeroing of other offsets, since they are constants in time:

$$D_y^{(ij)} D_x^{(ij)} + D_z^{(ij)} + d_z^{(ij)} \rightarrow 0.$$

Finally we have: $\Delta W_{ij} = \alpha \int_t^{t+\Delta t} a_j \widetilde{x}_i dt$ (here $\widetilde{x}_i$ is a signal on the output of the RC-circuit).

It is a very interesting question whether such neural network (with RC-circuits in the feedforward network) gives the correct results or not. Both the mathematical analysis and the computer simulations demonstrate very fast and stable convergence of *this type* of ANN (in comparison with ideal ANN)! It is possible to prove also that such modification of ANN has the same global minimum of error as an ideal ANN (with the same architecture, i.e. with the same number of neurons, interconnections and the same transfer functions).

3.7.5 Self-correction System of the Second Type

Let us consider now the Self-correction System of the Second Type, which implies that at any time the corrected (by this system) learning system is *precisely* providing the learning (the weights' modification) only for the current set of weights. That is, if in result of learning (which has followed after the self-correction procedure) the weights changed considerably it will affect the precision of the learning system.

The ideal learning system (for MLP ANN with backpropagation learning) provides weight modification in accordance with $\Delta W_{ij} = -\alpha \frac{\partial \mathcal{E}}{\partial W_{ij}}$, where α is a speed-of-learning coefficient, $\mathcal{E}$ is a mean-square-error. This, in particular, means, that if the *zero* signal is applied to the instantaneous error g input of the backward network (learning system) of the backpropagation-based system, *all the weights must stay unchanged.*

This property of ideal ANN was used in Self-correction System of the Second Type. The regime of the function of ANN, based on such a self-correction system, is the following: the learning process (when the weights are modified) is alternating with the

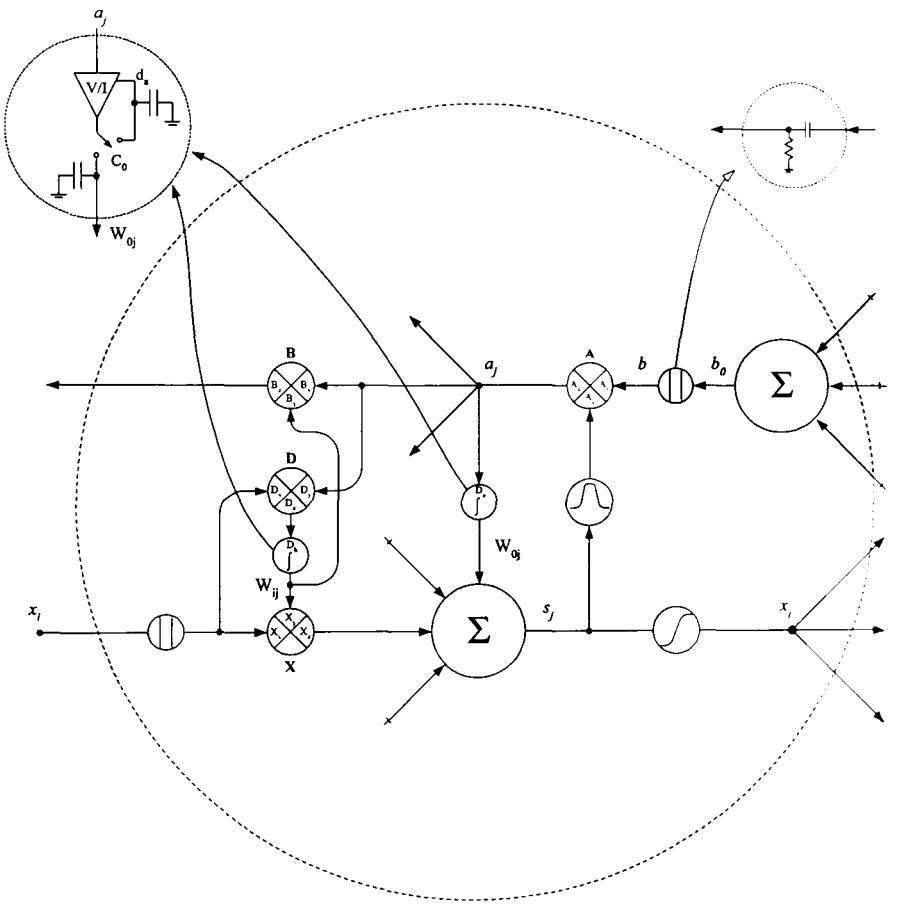


Figure 84. Schematic of ANN with Self-correction System of the First Type, Second Kind.

self-correction procedure (when the nonidealities of the learning system are suppressed) (for example - 10 cycles of learning alternating with 5 cycles of self-correction).

In the self-correction stage:

- the weights are fixed;
- the input signal is the same as during the learning procedure;
- the zero signal is applied on the instantaneous error g input;
- the voltage-to-current converters, responsible for the setup of the weights, are commutated to *their* correcting inputs (see Figure 85). The steady-state values d_{ij}, established as a result of the self-correction cycles (which actually provide the above mentioned *necessary condition* of the ideal learning system: $\Delta W_{ij} = 0$, if g=0) are to be fixed in the correcting capacitors.

Let us prove that the ANN, based on such a self-correction procedure, will tend to converge to the same global minimum of error as the ideal ANN with the same neural architecture. (As for the speed and the stability of the convergence, it is a subject for additional research).

For the learning system **during the learning stage** we have:

$$a_{out}^{(learn)} = d_k - x_k^{(out)};$$

$$a_j^{(learn)} = \left(f'(s_j) + A_y^{(j)}\right)\left(\sum_k \left(\left(W_{jk} + B_y^{(jk)}\right)\left(a_k^{(learn)} + B_x^{(jk)}\right) + B_z^{(jk)}\right) + A_x^{(j)}\right) + A_z^{(j)}; \tag{47}$$

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} \left(\left(a_j^{(learn)} + D_y^{(ij)}\right)\left(x_i + D_x^{(ij)}\right) + D_z^{(ij)} + d_{ij}\right) dt, \tag{48}$$

$$W_{ij} = W_{ij} + \Delta W_{ij};$$

$$x_j = f\left(\sum_i \left(\left(W_{ij} + X_y^{(ij)}\right)\left(x_i + X_x^{(ij)}\right) + X_z^{(ij)}\right) + W_j^0\right).$$

At the **self-correction stage** we are applying the *same signals* x_j on inputs of ANN and the *zero* signal on the input of the learning system $a_{out}^{(sc)} = 0$. The error signal propagation will be described by the equation

$$a_j^{(sc)} = \left(f'(s_j) + A_y^{(j)}\right)\left(\sum_k \left(\left(W_{jk} + B_y^{(jk)}\right)\left(a_k^{(sc)} + B_x^{(jk)}\right) + B_z^{(jk)}\right) + A_x^{(j)}\right) + A_z^{(j)}; \tag{49}$$

The dynamics of the correcting parameters d_{ij} are described by the following equation:

$$\Delta d_{ij} = \alpha \int_t^{t+\Delta t} \left(\left(a_j^{(sc)} + D_y^{(ij)} \right) \left(x_i + D_x^{(ij)} \right) + D_z^{(ij)} + d_{ij} \right) dt, \qquad (50)$$
$$d_{ij} = d_{ij} + \Delta d_{ij};$$

If the self-correction is fast enough (which is a normal situation, since the learning, usually, is very slow: $\Delta W_{ij} \ll W_{ij}$), we will have the steady state of correcting parameters d_{ij}, i.e. $\Delta d_{ij} = 0$. Substituting the expression for Δd_{ij} (Eq. 3.50), which are equal to zero, into Eq. 3.48 we will have:

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} \left(a_j^{(learn)} - a_j^{(sc)} \right) \left(x_i + D_x^{(ij)} \right) dt.$$

We can introduce a new variable $\widetilde{a}_j \equiv a_j^{(learn)} - a_j^{(sc)}$. Hence for the ΔW_{ij} we will have:

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} \widetilde{a}_j \left(x_i + D_x^{(ij)} \right) dt. \qquad (51)$$

Similarly, for ΔW_0 we will have:

$$\Delta W_j^0 = \alpha \int_t^{t+\Delta t} \widetilde{a}_j dt. \qquad (52)$$

Let us analyse now equation 3.51, and assuming that ANN state is near the steady-state, this means that $\Delta W_{ij}, \Delta W_j^0 \underset{t\to\infty}{\to} 0$. (The question is whether or not the steady-state error of imperfect ANN with the suggested self-correction system will be the same as in case of ideal ANN with the same architecture?) Taking into consideration equation 3.52, we will have

$$\Delta W_{ij} = \alpha \int_t^{t+\Delta t} \widetilde{a}_j \left(x_i + D_x^{(ij)} \right) dt \underset{t\to\infty}{\to} \alpha \int_t^{t+\Delta t} \widetilde{a}_j x_i dt. \qquad (53)$$

For the variable $\widetilde{a}_j \equiv a_j^{(learn)} - a_j^{(sc)}$, taking into account the Equations 3.47 and 3.49, we will have:

$$\begin{aligned} \widetilde{a}_j &= \left(f'(s_j) + A_y^{(j)} \right) \left(\sum_k \left(W_{jk} + B_y^{(jk)} \right) \left(a_k^{(learn)} - a_k^{(sc)} \right) \right) = \qquad (3.54) \\ &= \left(f'(s_j) + A_y^{(j)} \right) \left(\sum_k \left(W_{jk} + B_y^{(jk)} \right) \widetilde{a}_k \right). \end{aligned}$$

If the offsets $A_y^{(j)}$=0, then the equation (3.54) could be rewritten as

$$\widetilde{a}_j = f'(s_j) \left(\sum_k \left(W_{jk} + B_y^{(jk)} \right) \widetilde{a}_k \right). \qquad (55)$$

If we will introduce the new variables: $\widetilde{W}_{jk} \equiv W_{jk} + B_y^{(jk)}$, we will have finally:

$$\widetilde{a}_j \simeq f'(s_j) \left(\sum_k \widetilde{W}_{jk} \widetilde{a}_k \right). \tag{56}$$

For the feedforward network we will have:

$$x_j = f \left(\sum_i \left(\left(\widetilde{W}_{jk} - B_y^{(jk)} + X_y^{(ij)} \right) \left(x_i + X_x^{(ij)} \right) + X_z^{(ij)} \right) + W_j^0 \right).$$

As was stated in subsection 3.7.2, the offsets of the feedforward network weakly influence the dynamics and final error of ANN.

As for the learning system, the comparison of equations (3.56), (3.53) and (3.52) with equations (3.39), (3.40) and (3.41) respectively, allows us to conclude that the suggested system converge to the same global minimum as the ideal ANN (since the equations describing the dynamics of the suggested system coincide with the equations describing the ideal ANN, where $\widetilde{a}_j$ is an *efficient* back-propagating error signal whereas in case of ideal ANN it is a_j).

Therefore the suggested system is tolerant of each offset of multipliers **X, D** and **B**, as well as of offsets $A_x^{(j)}$ and $A_z^{(j)}$ of multipliers **A**.

The condition $A_y^{(j)} = 0$ essentially means that no signal should propagate in back direction through a particular neuron if it is saturated (thus $f'(s_j) = 0$). In our view it should not be a problem to implement such a condition in hardware, especially if we will take into consideration that:

1. in a case of standard MLP ANN with sigmoidal activation function, the $f'(s_j)$ is a nonnegative function.
2. the precision of the shape of function $f'(s_j)$ could be substantially different in comparison with the ideal derivative of the activation function $f(s_j)$ (this is also a subject for additional research).

Final notes: The considered self-correction system of the second type seems less attractive than e.g. the self-correction system of the first type (the one with RC-circuits in feedforward and in backward networks), since there is only one self-correction procedure, and this takes place before the learning (not a periodical self-correction during the learning as in the case of the self-correction of the second type). The situation, however, looks different if we take into consideration the nonlinear distortions of the learning system. It turns out that the self-correction system of the second type is more

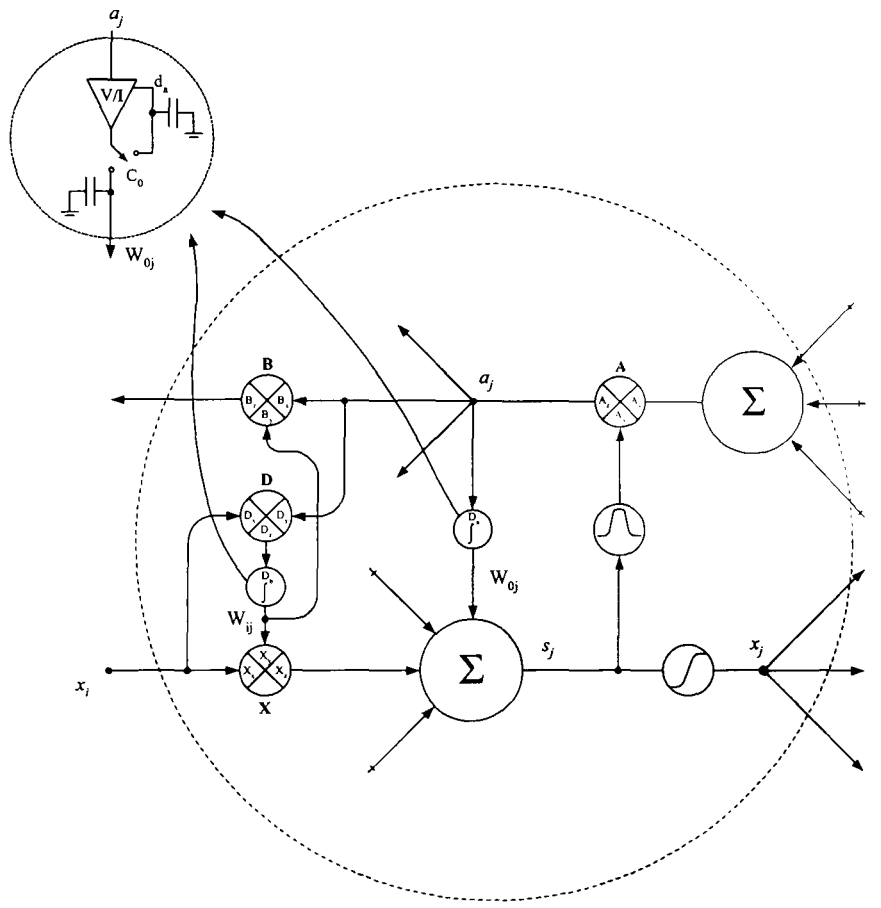


Figure 85. A schematic of ANN with Self-correction System of the Second Type.

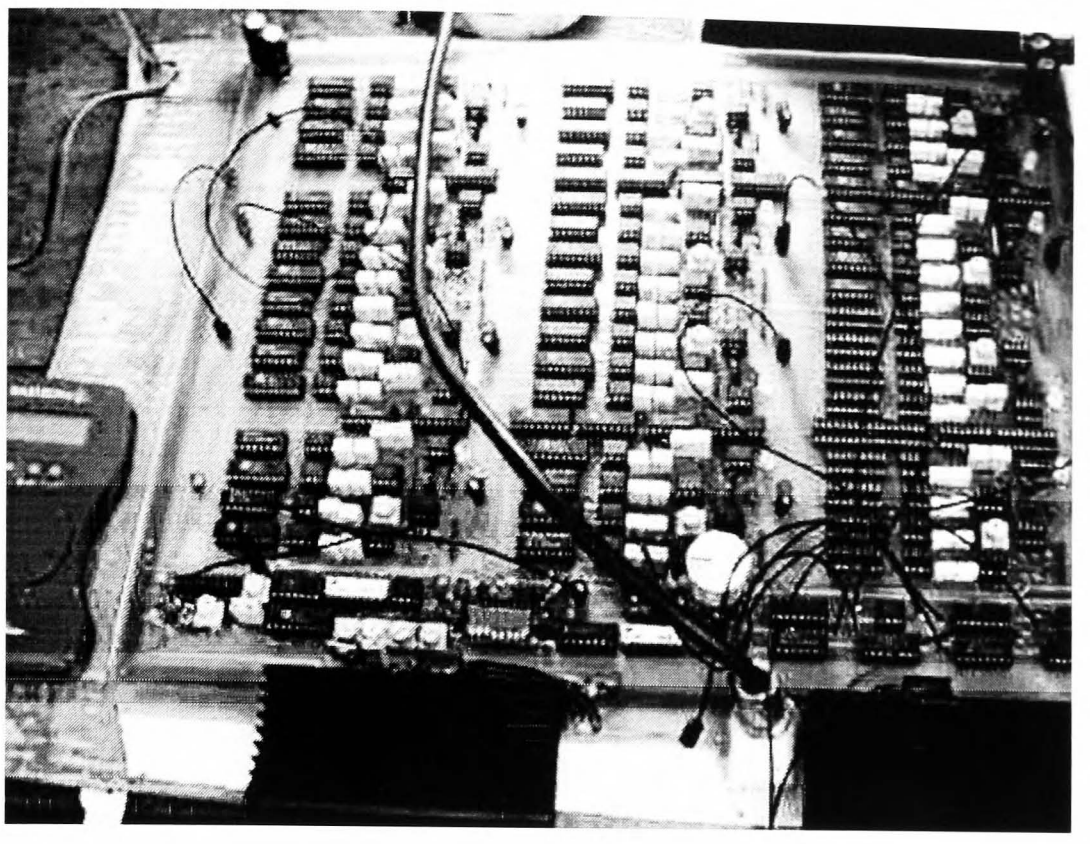


Figure 86. Breadboard of analogue ANN (with multilayer perceptron architecture) with on-board learning

tolerant of such kind of nonidealities of analogue systems. It looks like the addition of RC-circuits both in feedforward and in backward networks of the ANN with the self-correction of the second type would improve the precision characteristics of ANN.

Another important property of suggested ANN architectures is their tolerance of damage or degradation of their elements. The analysis, which is not included in the current work, shows that the ANNs based on self-correction of the second type are more tolerant of damage of elements in comparison with these based on self-correction of the first type. The RC-circuits both in feedforward and in backward interconnections of ANNs also increase their tolerance of damage or degradation of the elements of ANN.

3.7.6 Experiment

A breadboard of analogue ANN (with multilayer perceptron architecture) with on-board learning was created (see Figure 86). The breadboard is implementing the self-correction system of the second type.

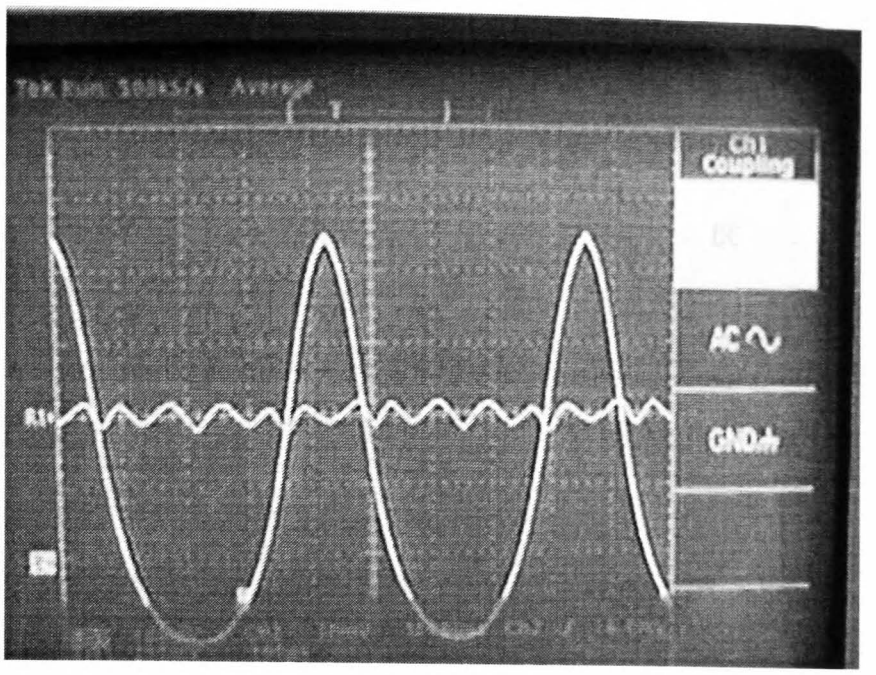


Figure 87. Example of approximation: the two signals (unresolvable on the screen of the oscilloscope) and the error function multiplied by 10.

The main unit of the system is the analogue multiplier MLT04 (Analog Devices : http://www.analog.com/pdf/mlt04.pdf), with precision about 1%. The frequency of the learning cycle repetition was within the range from 1 to 10 kHz.

The system is implementing the neural architecture 4x4x4x4, that is 3 layer neural network with 4 inputs and 4 fully interconnected neurons in each layer. Such an architecture allows in principle to solve the task of approximation of multivariable (up to 4 variables) functions. In particular, the example of the approximation task, considered in the above subsections (1x3x4x1) could be realised.

The system had demonstrated the ability to solve the task of approximation. The example of approximation is shown in Figure 87. The two signals - output of the ANN and the sample signal (unresolvable on the screen of the oscilloscope) as well as the error function, multiplied by 10 are shown.

The results of the experiment were mixed (partially encouraging, partially disappointing). On one hand, the precision of the units, to be precise, was very high (better than 10^{-7}). On the other hand, the precision of approximation was not high enough in comparison with the ideal ANN (with the same architecture, solving the same task). The reason seems to be the imperfection of the multipliers **A** (in particular, the unsuppressed offsets $A_y^{(j)} = 0$). It turned out that the learning system is very sensitive to these offsets (and this had been underestimated).

3.7.7 Conclusions

Let us briefly summarise the results of this section:

- Supervised-learning ANN (like MLP) with parallel learning must have a very precise learning system, whereas the feedforward network could be relatively imprecise (the influence of imprecision is a subject for additional research).
- The "local" self-correction system (described in 3.7.3) is relatively complex and doesn't give the maximal possible precision.
- The "Global" self-correction system is more precise than the "Local" one, because:

  1. it minimizes the harm caused by the residual error of elements of the learning system

  2. it suppresses the error contributed by the interconnections;
- "Global" self-correction systems are also simpler than "local";
- The additional RC circuits added to the *feedforward* network of ANN with self-correction system (of the first type, second kind, see the Fig. 84) will influence on the behaviour of ANN (the learning dynamics), but ANN will converge to **the same** global minimum of error as the **ideal** ANN. Therefore such an additional (RC circuits) to the neural scheme is a subject for the further research even **in case of standard ANN** (whether RC-circuits will improve the learning (making it faster and more stable) or will worsen the learning characteristics of ANN?).
- The additional RC circuits added to the *learning* network of ANN with self-correction system of the first type (see Fig. 82 an Fig. 84 will influence on the behaviour of ANN (the learning dynamics), but ANN will converge to **the same** global minimum of error as the **ideal** ANN. Therefore such an additional (RC circuits) to the neural scheme is a subject for the further research even **in case of standard ANN** (whether RC-circuits will improve the learning (making it faster and more stable) or will worsen the learning characteristics of ANN?).

- All the considered self-correction procedures were analysed in assumption of linearity of analogue multipliers. The time delay effects also were out of scope of the current consideration.
- In addition to the ability of the suggested systems to suppress the offsets, the "global" self-correction system-based ANN seem to be tolerant of damage and degradation of their elements (this question is also a subject for additional research).
- Such self-correction systems are necessary for creation of the very computationally powerful purely analogue ANN on the basis of Ultra Large Scale Integrity technologies. Such ANNs are necessary in particular for creation of systems of artificial intelligence.

Chapter 4
Conclusions

4.1 Main conclusions

- *Analogue* technology will play a very important role in neural science and in information processing technology both in the foreseeable future (creation of compact, low power consuming information processing systems, such as systems of image/sound processing) and in the long-term outlook (creation of systems of artificial intelligence).
- Analogue *polynomial* adaptive systems could be used in different areas (for calibration, nonlinear information processing, etc.) in the near future.
- The principle of *orthogonality* must be widely employed by analogue information processing systems. The suggested architecture of the Synthesiser of Orthogonal Functions allows the creation of complex purely analogue information processing systems.
- The use of special *self-correction* systems and procedures is necessary for creation of complex high precision analogue systems of information processing (in particular, neural networks). Such self-correction systems and procedures could be simpler than the learning system. On the other hand, the procedure of self-correction should not be trivial (that is just self-correction of separate elements).
- Another important property of analogue neural networks is their potential high tolerance to degradation of the elements (or parts of the system).

4.2 Conclusions of different sections of the Thesis

4.2.1 Conclusions of the section "Introduction":

Artificial systems of information processing could be much more computationally efficient than natural ones. The aim of my work is to contribute to the creation of complex and supercomplex computational systems, in particular ANN, which, supposedly, will be the main components of Artificial Intelligence systems. Another aim is to suggest simple and highly efficient computational systems, which could be implemented in the near future.

The computational power of artificial systems could become comparable with that of the human brain in the foreseeable future. The creation of systems of artificial intelligence however is a much greater problem since firstly the proper computational architecture has to be established and secondly, the problem of huge (~10^5 Gbyte) memory of the AI system must be solved in hardware.

A special self-correction system is necessary for creation of highly efficient analogue neuromorphic systems.

The low power consumption of analogue ANN gives the opportunity of creating the multilayer integrated structure, thus of increasing the scale of integration by several (more than four) orders of magnitude.

Polynomial-based adaptive analogue systems as well as adaptive systems utilising the principle of orthogonality, could be very efficient and could find commercial applications already in the near future.

In many typical ANN tasks (especially in cases of complex tasks) multi-layer perceptron neural architecture could be much more efficient than polynomial-based adaptive systems. In many cases the principle of orthogonality is not applicable.

4.2.2 Conclusions of the section "Analogue Polynomial Approximator":

The suggested systems performing functions of approximation, interpolation and extrapolation, were studied experimentally and by methods of computer modeling.

The devices showed good working characteristics (corresponding to theoretical description) In my opinion they can find application in high precision analogue engineering for fast calibration of different devices, as an adaptive nonlinear element capable of compensating nonlinearity of radio-engineering units, etc.

Such systems could be pedagogically interesting as simplest adaptive neuromorphic systems. The Legendre polynomials' based approximator is a good demonstration of neuromorphic system using the principle of orthogonality.

4.2.3 Conclusions of the section "Synthesiser of orthogonal functions"

It is possible to produce on the basis of analogue elements a plurality of mutually orthogonal signals such as Legendre Polynomials, Chebyshev Polynomials, Cosine Basis of Functions, Smoothed Cosine Basis, etc.

The recurrent equations (3.30), where a_i are defined by integral equations (3.31) allow to produce a basis of orthogonal functions $F_i(x)$ if the functions $f(x)$ and $g(x)$ applied at the inputs of the synthesiser are respectively odd and even

The condition Eq.(3.31 could be realised by means of the simple feedback shown in the Fig. 39(a).

The process of the synthesis of orthogonal functions is fast and very stable.

In the case of high precision and high orthogonality requirements, a scheme of an Analogue Synthesiser of orthogonal signals must include some additional units for offset zeroing and normalising in addition to the scheme shown in Fig. 39.

In the case not the highest requirements of precision and orthogonality it is possible to create the Synthesiser with *not-adapting* but with *pre-set* (in analogue or digital memory) coefficients a_i. In this case the Synthesiser will be a little simpler and will not demand a special phase of adaptation but will be less precise as well as more sensitive to temperature variation and to degradation of elements over time.

The maximal frequencies of function of the Synthesiser implemented on the basis of up-to-date analogue VLSI technology working are supposed to be several hundreds megahertz. But in addition to the problem of off-sets compensation in a high frequency range, the problem of an inter-cascade delay of signals must be solved.

4.2.4 Conclusions of the section "Analogue parallel-learning MLP neural network"

Supervised-learning ANN (like MLP) with parallel learning must have a very precise learning system, whereas the feedforward network could be relatively imprecise (the influence of imprecision is a subject for additional research).

The "local" self-correction system (described in 3.7.3) is relatively complex and doesn't give the maximal possible precision.

The "Global" self-correction system is more precise than the "Local" one, because:

1. it minimizes the harm caused by the residual error of elements of the learning system
2. it suppresses the error contributed by the interconnections;

"Global" self-correction systems are also simpler than "local";

The additional RC circuits added to the *feedforward* network of ANN with self-correction system (of the first type, second kind, see the Fig. 84) will influence on the behaviour of ANN (the learning dynamics), but ANN will converge to **the same** global minimum of error as the **ideal** ANN. Therefore such an additional (RC circuits) to the neural scheme is a subject for the further research even **in case of standard ANN** (whether RC-circuits will improve the learning (making it faster and more stable) or will worsen the learning characteristics of ANN?).

The additional RC circuits added to the *learning* network of ANN with self-correction system of the first type (see Fig. 82 an Fig. 84 will influence on the behaviour of ANN (the learning dynamics), but ANN will converge to **the same** global minimum of error as the **ideal** ANN. Therefore such an additional (RC circuits) to the neural scheme is a subject for the further research even **in case of standard ANN** (whether RC-circuits will improve the learning (making it faster and more stable) or will worsen the learning characteristics of ANN?).

All the considered self-correction procedures were analysed in assumption of linearity of analogue multipliers. The time delay effects also were out of scope of the current consideration.

In addition to the ability of the suggested systems to suppress the offsets, the "global" self-correction system-based ANN seem to be tolerant of damage and degradation of their elements (this question is also a subject for additional research).

Such self-correction systems are necessary for creation of the very computationally powerful purely analogue ANN on the basis of Ultra Large Scale Integrity technologies. Such ANNs are necessary in particular for creation of systems of artificial intelligence.

Appendix A.
The Proof of Orthogonality of functions generated by the Synthesiser described in 3.2

Let us suggest that $f(t)$ and $g(t)$ are some respectively odd and even functions and are defined on an interval $(-1, 1)$. Let's define functions $F_i(t)$ as

$$\begin{aligned} F_0(t) &= g(t); \\ F_1(t) &= f(t)g(t); \\ F_2(t) &= f(t)F_1(t) + a_2F_0(t); \\ &\cdots \\ F_i(t) &= f(t)F_{i-1}(t) + a_iF_{i-2}(t), \end{aligned} \tag{1}$$

where a_i for $i \geq 2$ is defined from equation:

$$\int_{-1}^{1} F_i(t)F_{i-2}(t)dt = 0. \tag{2}$$

It is possible to prove the following

Proposition 1 Functions $F_i(t)$ are mutually orthogonal on an interval $(-1, 1)$, that is: $\int_{-1}^{1} F_i(t)F_j(t)dt = 0$ for any i and any j: $i \neq j$

Proof. Firstly, taking into consideration that the product of two odd functions is an even function and that the product of odd and even functions is an odd function it is easy to see that $F_i(t)$ is even function if i is even number, and $F_i(t)$ is odd function if i is odd number. So, taking into consideration that any even function is orthogonal to any odd function on the range $(-1, 1)$, it is necessary to prove only that every even function is orthogonal for every even function and that every odd function is orthogonal for every odd function. Besides in accordance with Eq.(A.2): F_i is orthogonal to F_j if $j = i \pm 2$ (for the sake of simplicity we will write F_i, f, g instead of $F_i(t), f(t), g(t)$). Therefore, to prove the proposition it is enough to prove that at $\forall i$: F_i is orthogonal to F_{i-2k} where k is $i \geq 2k \geq 4$ (so if i is an odd number then $i > 2k \geq 4$).

Let's prove that F_4 is orthogonal to F_0 and that F_5 is orthogonal to F_1.

$$\int_{-1}^{1} F_4F_0dt = \int_{-1}^{1} (fF_3 + a_4F_2)\, F_0dt = \int_{-1}^{1} (F_3F_1 + a_4F_2F_0)\, dt = 0. \tag{3}$$

Here we have used the definition of F_i Eq.(A.1), so: $F_4 = fF_3 - a_4F_2$ and $fF_0 = F_1$. The last integral is equal to zero in accordance with Eq.(A.2).

Analogously:

$$\int_{-1}^{1} F_5F_1dt = \int_{-1}^{1} (fF_4 + a_5F_3)\, F_1dt = \int_{-1}^{1} (F_4F_2 + a_5F_3F_1)\, dt = 0.$$

Thus for $\forall i, j \leq 5$ and $i \neq j$ the proposition is valid.

Now, let's prove that if for some integer number I $(I \geq 5)$ and for $\forall i : i \leq I$ and $\forall k : i \geq 2k \geq 4$ (so if i is an odd number then $i > 2k \geq 4$) F_i is orthogonal to F_{i-2k} :

$$\int_{-1}^{1} F_iF_{i-2k}dt = 0 \tag{4}$$

then F_{I+1} will be orthogonal to F_{I+1-2k} for any k so that $I + 1 \geq 2k \geq 4$:

$$\int_{-1}^{1} F_{I+1}(t)F_{I+1-2k}(t)dt = 0.$$

The same statement it is possible to formulate as:

if we have a basis of first $I + 1$ functions F_i where $I \geq i \geq 0$, defined by equations (A.1) and (A.2) and these functions are mutually orthogonal;

then F_{I+1} defined by equations (A.1) and (A.2) will orthogonal to any of the first F_i functions $(I \geq i \geq 0)$.

If we prove this statement, in accordance with the principle of mathematical induction we will prove the proposition.

Let us expand the expression for a inner product coefficients between $(I + 1) - th$ function and $(I + 1 - 2k) - th$ functions (using Eq.(A.1)):

$$\int_{-1}^{1} F_{I+1}F_{I+1-2k}dt = \int_{-1}^{1} (fF_I + a_{I+1}F_{I-1})\, F_{I+1-2k}dt \tag{5}$$

$$= \int_{-1}^{1} \begin{pmatrix} F_I\,(fF_{I+1-2k}) \\ +a_{I+1}F_{I-1}F_{I+1-2k} \end{pmatrix} dt = \int_{-1}^{1} \begin{pmatrix} F_IF_{I+2-2k} \\ -a_{I+2-2k}F_IF_{I-2k} \\ +a_{I+1}F_{I-1}F_{I+1-2k} \end{pmatrix} dt$$

It can be seen that *all terms in Eq.(A.5) are* ***non-diagonal*** *elements* of an inner product matrix C_{ij} of functions F_i $(I \geq i \geq 0)$. Taking into considaration the orthogonality of F_i $(I \geq i \geq 0)$ basis of functions (in accordance with the condition (A.4)) and hence $C_{ij} \equiv \int_{-1}^{1} F_i(t)F_j(t)dt = \delta_{ij}$ we can conclude that for any k so that $I + 1 > 2k \geq 4$ all terms in Eq.(A.5) are equal zero.

Special case: when $2k = I + 1$ (obviously possible only in cases of odd I). In this case the second term in last integral of Eq.(A.5) is proportional to $\int_{-1}^{1} F_I F_{I-2k} dt = \int_{-1}^{1} F_I F_{-1} dt$, where the function F_{-1} is not defined. So, we should prove that if $2k = I + 1$, then $\int_{-1}^{1} F_{I+1} F_{I+1-2k} dt = \int_{-1}^{1} F_{I+1} F_0 dt = 0$:

$$\int_{-1}^{1} F_{I+1} F_0 dt = \int_{-1}^{1} \left(f F_I + a_{I+1} F_{I-1}\right) F_0 dt = \int_{-1}^{1} \left(F_I F_1 + a_{I+1} F_{I-1} F_0\right) dt. \quad (6)$$

Here were used Eq.(A.1): $F_1 = fg = fF_0$. It is easy to see that all terms in the last integral of Eq.(A.6) are equal to zero in accordance with the precondition (A.4). Therefore for any k so that $I + 1 \geq 2k \geq 4$: $\int_{-1}^{1} F_{I+1} F_{I+1-2k} dt = 0$ ■

References

1. J. Dayhoff, *Neural Network Architectures-An Introduction*, Van Nostrand Reinhold, New York, 1990.

2. A. Murray and L. Tarassenko, *Analogue Neural VLSI: A Pulse Stream Approach*, Chapman and Hall, 1994.

3. J. Dowling, *The Retina: An Approachable Part of the Brain*, MA: Belknap Press of Harvard University Press, 1987.

4. J. Von Neuman, *Theory of Self-Reproducing Automata*, University of Illinois Press, 1966.

5. S. Haykin, *Neural Networks: A Comprehansive Foundation*, Maxwell Macmillian Canada, Inc., 1994.

6. C. Mead, *Analog VLSI and Neural Systems*, Addson-Wesley, Reading, MA, USA, 1989.

7. G. Luger and W. Stubblefield, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, The Benjamin/Cummings Publishing Company, Inc., 1993.

8. A. M. D. Poole and R. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, 1998.

9. C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE* **78**(10), pp. 1629–1636, 1990.

10. A. N. Kolmogorov, "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition," *J Doklady Akad. Nauk. SSSR* **114**, pp. 369–373, 1957.

11. G. P. Liu, V. Kadirkamanathan, and S. A. Billings, "On-line identification of nonlinear systems using volterra polynomial basis function neural networks," *Neural Networks* **11**(9), pp. 1645–1657, 1998.

12. J.-T. Lee, Tsu-Tian; Jeng, "Chebyshev polynomials-based (CPB) unified model neural networks for function approximation," *Proc. SPIE* **3077**, pp. 372–381, 1997.

13. V. N. Chesnokov, "Fast training analog approximator on the basis of legendre polynomials," *Proc. SPIE* **3077**, pp. 382–387, 1997.

14. M. A. Arbib, *Brains, Mashines and Mathemetics*, Springer Verlag, New York, 2nd ed., 1987.

15. W. S. McCulloch and W. H. Pitts, "A local calculus of the ideas imminent in nervous activity," *Bull Math. Biophy.* , pp. 115–133, 1943.

16. S. I. Amari, "Mathematical foundations of neurocomputing," *Proceedings of the IEEE* , pp. 1443–1463, 1990.

17. R. D. E. McClelland, T. L. and the PDP Research Group., *Parallel Distributed Processing*, The MIT Press, 1986.

18. F. J. Pineda, *Generalization of Backpropagation to Recurrent and Higher Order Neural Networks*, American Institute of Physics, New York, 1988.

19. F. Rosenblatt, *Principles of Neurodynamics*, Spartan Books, Washington, DC, 1962.

20. B. Widrow, *Generalization and Information Storage in Networks of Adaline "neurons"*, Sparta, 1962.

21. B. Widrow and M. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proceedings of the IEEE* **78**, pp. 1415–1442, 1990.

22. P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences*. PhD thesis, Harward University, Cambridge, MA, 1974.

23. H. G. E. Rumelhart, D. E. and R. J. Williams, "Learning representations by back-propagating error," *Nature (London)* , pp. 533–536, 1986.

24. D. B. Parker, "Learning-logic: Casting the cortex of the human brain in silicon," tech. rep., Center for Computational Research in Economica and Management Science, MIT, Cambridge, MA, 1985.

25. M. L. Minsly and S. A. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

26. B. Gilbert, "A precise four-quadrant multiplier with subnanosecond response.," *IEEE Journal of Solid-State Circuits* , 1968.

27. O. H. Schmitt, "An electrical theory of nerve impulse propagation," *Am. J. Physiol.* , pp. 399–400, 1937.

28. P. Horowitz and W. Hill, *The Art of Electronics*, Cambridge University Press, UK, 1989.

29. H. Stark and F. B. Tuteur, *Electrical Communication - Theory and Systems*, Prentice Hall, Englewood Cliffs, NJ, 1979.

30. A. F. Murray and A. V. W. Smith, "Asynchronous arithmetic for VLSI neural systems," *Electronics Lett.* , pp. 642–643, 1987.

31. A. F. Murray and A. V. W. Smith, "A novel computational and signalling method for VLSI neural networks," *European Solid State Circuits Conference* , pp. 19–22, 1987.

32. H. McCartor, *Back Propagation Implementation on the Adaptive Solutions CNAPS Neurocomputer Chip*, Morgan Kaufmann Pub., 1991.

33. Y. Deville, "Digital VLSI neural networks: From versatile neural processors to application-specific chips," *ICANN'95, Oct. 9-3,* , 1995.

34. T. S. C. H. Holler, M. and R. Benson, "An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses.," in *International Joint Conference on Neural Networks (IJCNN89)*, pp. 191–196, 1989.

35. B. Widrow and M. E. Hoff, "Adaptive switching circuits," *IRE Western Electric Show and Convention Record* , pp. 94–104, 1960.

36. R. J. C. M. A. Jabry and B. G. Flower, *Adaptive Analogue VLSI Neural Systems*, Chapman and Hall, 1996.

37. J. J. P. A. J. Montalvo, R. S. Gyurcsik, "Toward a general-purpose analog VLSI neural network with on-chip learning," *IEEE Transactions on Neural Networks* **8**(2), pp. 413–423, 1997.

38. W. K. Pratt, *Digital Image Processing*, Chichester: Wiley, New York, 1978.

39. K. B. Datta and B. M. Mohan, *Orthogonal Functions in Systems and Control*, World Scientific Pub Co, 1995.

40. F. K. S. Chi-Shi Liu, Hsiao-Chuan Wang and C.-S. Huang, "An orthogonal polynomial representation of speech signals and its probabilistic model for text independent speaker verification," *Proc. ICASSP* , pp. 345–348, 1995.

41. P. D. A. N. Akansu and X. Lin, "Orthogonal transmultiplexers in communication: A review,," *IEEE Trans. Signal Processing* , pp. 979–995, 1998.

42. M. Abramowitz, *Handbook of Mathematical Functions with Formulae, Graphs and Mathematical Tables*, Wiley-Interscience, New York; Chicester, 1972.

43. J.-T. Lee, Tsu-Tian; Jeng, "Chebyshev polynomials based (CPB) unified model neural networks for function approximation," *Proc. SPIE* **3077**, pp. 372–381, 1997.

44. V. N. Chesnokov, "Fast training analog approximator on the basis of legendre polynomials," *1996 IEEE Intern. Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing* , pp. 299–304, 1996.

45. V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine* **8**(3), pp. 10–26, 1991.

46. R. D. Nowak and B. D. V. Veen, "Tensor product basis approximations for volterra filters," *IEEE Transactions on Signal Processing* **44**, pp. 36–50, January 1996.

47. D. Hecht *IEEE Transaction of Sonocs and Ultrasonics* , pp. 7–17, 1977.

48. V.N.Chesnokov and V.V.Proklov, "Peculiarities of multifrequency acoustooptical interaction in resonant photoelastic materials," *Solid State Physics (Russian)* , pp. 3268–3279, 1994.

49. V.N.Chesnokov and V.V.Proklov, "AO intermodulation effect in resonant photoelastic materials," *Proc. SPIE 2643* , pp. 253–256, 1995.

50. Z. R. D. J. Jobson and G. A. Woodell, "Properties and performance of a Center/Surround retinex,," *IEEE Transactions on Image Processing* , March 1997.

51. R. S. G. Antonio J. Montalvo and J. J. Paulos, "Toward a general-purpose analog VLSI neural network with on-chip learning," *IEEE Transaction on Neural Networks* **8**(2), pp. 413–423, 1997.

52. M. L. D. Andes, B. Widrow and E. Wan, "MRIII: A robust algorithm for training neural networks," *Proc. Int. Joint Conf. Neural Networks* , pp. 553–556, 1990.

53. M. Jabri and B. Flower, "Weight perturbation: Anoptimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks," *IEEE Transaction on Neural Networks* , pp. 154–157, 1992.